Universidad Autónoma de Madrid

Escuela politécnica superior

Trabajo de Fin de Grado

IMPLEMENTACIÓN Y VALIDACIÓN DE UN SISTEMA ELECTRÓNICO DE FACTURACIÓN DE CLOUD SERVICES

Alma Castillo Antolin

Tutor: Javier Lucio Ruiz-Andino

Ponente: Miren Idoia Alarcón Rodríguez

Junio 2014

IMPLEMENTACIÓN Y VALIDACIÓN DE UN SISTEMA ELECTRÓNICO DE FACTURACIÓN DE CLOUD SERVICES

AUTOR: Alma Castillo Antolin
TUTOR: Javier Lucio Ruiz-Andino
PONENTE: Miren Idoia Alarcón Rodríguez

Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2014

Resumen

Resumen

El objetivo de este proyecto es la definición de un modelo para la tarificación de servicios web y la implementación del mismo dentro de una tienda de servicios. Para ello, se han analizado los modelos teóricos y sistemas implementales de este tipo que existen en la actualidad y los distintos modelos de precio habitualmente aplicados a estos servicios.

Se ha definido un lenguaje de modelado de servicios y modelos de precio y se ha creado una interfaz de usuario que genera modelos en este lenguaje a partir de indicaciones del usuario. Se ha definido también un formato para la representación de los datos de usos de estos servicios y se han definido e implementado dos algoritmos de particionamiento temporal y facturación de dichos datos. El resultado del algoritmo de facturación es un fichero de salida con el desglose del precio a pagar y un archivo PDF de visualización del mismo. Asimismo, se ha implementado un componente que automatiza dicha facturación dado un periodo de tiempo.

Por último, se ha verificado, validado y evaluado el sistema resultante, tanto de forma individual como colectiva junto con usuarios potenciales del mismo.

Palabras Clave

Tarificación, modelado de precio, tarificación dinámica, algoritmo de facturación.

Abstract

The aim of this project is the definition of a web services pricing model and its implementation within a services' marketplace. For that purpose, theoric models and implemented systems of this kind as well as different pricing models usually applied to web services have been analysed.

A service and pricing modelling language has been defined and a user interface generating these models from user instructions has been built. A usage data representation format has been defined and two temporarily partitioning and data billing algorithms have been defined and implemented. The outcome of the billing algorithm is an output file including the itemized amount to pay and a PDF file to visualize it. An automatic billing component has also been implemented. It automates the billing process given a period of time.

Finally, the resulting system has been verified, validated and evaluated, individually and jointly with some of its potential users.

Key words

Billing, price modelling, dynamic pricing, billing algorithm.

Agradecimientos

Quiero agradecer a todas las personas que han participado en el desarrollo de este TFG. A Idoia, que me ha apoyado desde el principio, por ayudarme con los problemas que han surgido, atenderme siempre e implicarse en que todo saliera perfecto. A Alfonso, por escucharme divagar y pensar en voz alta, y por ayudarme a darle forma al proyecto y a tomar algunas de las decisiones más importantes. A toda la gente de TID: a mi tutor Javier Lucio, Luismi y Gonzalo por el apoyo; a Alberto por socorrerme cada vez que tenía una duda, a Jesús por la ayuda mutua y a Antonio por guiarme al aterrizar. Y, por supuesto, al grupo de becarios por los buenos ratos.

También quiero dar gracias a toda mi familia, especialmente a mi madre, por apoyarme durante estos años de estudio e impulsarme hasta aquí y hacia lo que viene después. Gracias también a quienes me han ayudado en los momentos de bajón a coger fuerzas y seguir con ganas, por motivarme a esforzarme más y más. Ha merecido la pena. Gracias a los amigos con quienes he pasado muchos buenos ratos durante estos 5 años. Y gracias también a aquellos que han contribuido en los proyectos futuros. GRACIAS

Índice general

Índice de figuras

1

Introducción

1.1. Ámbito del proyecto

Este Trabajo de Fin de Grado, realizado en colaboración con Telefónica I+D, está enmarcado dentro del proyecto de investigación TESLA. Dicho proyecto, subvencionado por la Junta de Castilla y León, tiene como propósito la implementación de una plataforma global de distribución de servicios de telecomunicaciones basados en la red de Telefónica. Además, ofrece capacidades como el control de llamadas, la mensajería, la localización o la comunicación M2M (Machine to machine). El proyecto TESLA cuenta con un demostrador denominado Marketplace que es una tienda de aplicaciones construidas sobre estas capacidades.

El Marketplace del proyecto TESLA requiere de un sistema de gestión de estos servicios que permita ofertar los mismos a clientes. Además, debe realizarse una facturación de dichos servicios a los usuarios en función del uso de los mismos.

1.2. Motivación del proyecto

En el contexto anteriormente descrito, es necesario un sistema de tarificación de aplicaciones y servicios de Internet. Al tratarse de servicios de Internet y a diferencia de otros productos comerciales, es posible obtener información de uso de los mismos. Por tanto, la tarificación realizada no sólo debe depender de la adquisición de productos sino de otros factores como el

uso realizado de los mismos. Además, pueden tenerse en cuenta factores como la calidad del servicio ofrecido o la relación entre la calidad ofrecida y la calidad comprometida.

Dicho sistema debe permitir modelar los servicios que se van a ofertar así como el precio asignado a cada uno de ellos. Estos modelos deben ser flexibles y adaptarse a distintos tipos de servicios de Internet. Además, los precios deben modelarse de forma que representen las distintas estrategias de precio aplicadas habitualmente a este tipo de servicios, teniendo en cuenta el uso del servicio y la calidad del mismo.

Actualmente, como se verá en la sección 3, no existen tarificadores de aplicaciones y servicios de Internet con estas características.

1.3. Objetivos y resultados finales

Por los motivos mencionados en la sección anterior, el objetivo de este Trabajo de Fin de Grado es diseñar, implementar y validar un sistema tarificador de servicios de Internet. Para ello debe, en primer lugar, definirse un lenguaje genérico de modelado de servicios y representación de modelos de precio. Este lenguaje debe ser flexible y adaptarse a los diversos modelos de precio usados habitualmente para la facturación de servicios de Internet.

Además, este sistema debe proporcionar una facturación para cada servicio contratado por un cliente. Esta facturación no debe depender únicamente del servicio en sí sino que debe de tener en cuenta tanto el uso que se ha hecho del mismo como la calidad del servicio ofrecido.

El resultado de este Trabajo de Fin de Grado es, por tanto, un sistema de definición de precios y tarificación de servicios de Internet íntegramente definido, diseñado, verificado y validado por la estudiante. Este sistema permite definir servicios y modelos de precios complejos mediante un lenguaje legible para el ser humano. Dicho lenguaje, basado en XML, es flexible y permite definir modelos de precio por tramos de manera genérica. Los modelos de precio definidos están basados en periodos de facturación tras los cuales se genera una factura que toma en consideración el uso realizado durante el mismo. Además, los modelos de precio tienen en cuenta la calidad del servicio ofrecido respecto al compromiso establecido. También permiten establecer descuentos por incumplimiento del acuerdo de nivel de servicio.

Asimismo, el sistema gestiona la contratación de los servicios definidos por parte de clientes y la generación de facturas periódicas automáticas para los mismos. El resultado de estas facturas depende del uso del servicio y de su modelo de precio. La contratación de cada servicio tiene una duración definida. El modelo de precio también puede definir cobros por la cancelación prematura del mismo.

Además, el sistema cuenta con una interfaz gráfica de usuario que gestiona las actividades de definición de servicios y modelos de precio, contratación, obtención de facturas, visualización y cancelación de contratos y visualización y borrado de productos.

Este sistema se compone, por tanto, de un modelo genérico de representación de modelos de precio y un algoritmo de tarificación sobre dichos modelos y unos datos de uso. Dicho modelo, compatible con la mayoría de modelos de precio usados actualmente en la tarificación de Cloud Services, se ha implementado e integrado dentro del Marketplace del proyecto TESLA y se ha integrado con funcionalidad de definición de servicios y gestión de contratos sobre los mismos. Se ha recubierto con una interfaz de usuario y una representación visual de las facturas obtenidas.

1.4. Estructura del documento

El resto de esta memoria se estructura de la siguiente forma:

En la sección 2 se detallan los objetivos de este Trabajo de Fin de Grado de manera más extensa. En la sección 3 se hace un análisis del estado del arte. Dicho análisis comprende tanto un análisis de los sistemas de tarificación, teóricos o prácticos, existentes, como un análisis de los distintos modelos de precio existentes en el mercado actual y su aplicabilidad al modelado de precios para Cloud Services.

En la sección 4 se define la aplicación que se va a diseñar, así como la metodología y las herramientas utilizadas. En la sección 5 se describen los requisitos funcionales y no funcionales de esta aplicación.

En la sección 6 se define el diseño del modelo a implementar. Para ello se describe detalladamente la arquitectura del mismo y se detalla la funcionalidad de sus componentes. Se describen también las distintas estructuras de datos a utilizar. En la sección 7 se documenta la implementación del modelo descrito. Para ello, se determinan las herramientas utilizadas para el desarrollo de cada componente y se presentan diagramas que muestran la estructura del código de la aplicación. Asimismo, se describen algoritmos de gran importancia para el proceso de tarificación.

En la sección 8 se muestra un caso real de servicio de Internet. Para este servicio se muestra todo un proceso de definición de producto y modelo de precio, contratación y tarificación. En la sección 9 se describe el proceso de verificación y validación de la aplicación construida para la comprobación del correcto funcionamiento de la misma. En la sección 10 se evalúa el resultado final y se compara con otros sistemas existentes en el mismo ámbito.

En la sección 11 se exponen las conclusiones obtenidas tras el desarrollo de este proyecto. Por último, en la sección 12, se plantean posibles líneas de trabajo futuro sobre este modelo de tarificación.

2

Objetivo

Como se ha especificado en el capítulo anterior, el objetivo de este Trabajo de Fin de Grado es la definición, implementación y validación de un sistema de tarificación de servicios de Internet. Para ello, se definen otros objetivos secundarios necesarios para el cumplimiento del objetivo principal. A continuación se detallan de forma más específica estos objetivos.

2.1. Lenguaje de definición y modelado

El sistema permite representar productos, modelos de precio y contratos. Para ello, se define un lenguaje específico, legible y editable por un ser humano, que permite representar estos conceptos de manera genérica.

Un mismo servicio o producto puede componerse de varias funcionalidades. El lenguaje de modelado permite definir todas estas funcionalidades como parte del mismo producto. Cada una de las funcionalidades ofrecidas por un producto puede contar además con varias unidades de medida. Estas unidades permitir cuantificar el uso de dicha funcionalidad que realice el usuario. Además, cada funcionalidad puede contar con diversos niveles o categorías de calidad de una determinada característica.

Para cada producto, debe definirse un único modelo de precio. Dicho modelo puede contar con varias opciones de contratación a las que denominamos como planes. Estas opciones se seleccionan de forma excluyente, de modo que en la contratación de un producto debe elegirse

exactamente una. Para cada uno de estos planes se define un conjunto de tipos de moneda y países para los que está disponible, de modo que no sea posible su contratación por parte de aquellos individuos no pertenecientes a dicho país o usuarios de dicha moneda. Cada plan define, además, un precio fijo a facturar en el momento de la contratación, un periodo de cobro, un precio periódico (que puede variar en función del periodo) y un conjunto de precios por uso. Estos precios por uso pueden ser incondicionales o depender del cumplimiento de ciertas condiciones acerca de la cantidad o el momento de uso. También pueden definirse valores máximo y mínimo para el total de cada periodo de cobro.

Cada plan de precio define también una duración para la contratación del mismo. Asimismo, puede definirse una penalización por la cancelación temprana que puede componerse de una cantidad fija y una cantidad dependiente del tiempo restante de contrato.

Las distintas calidades definidas para el producto pueden ofertarse dentro de un plan de precio. Cada una de estas ofertas está asociada a un nuevo precio agregado al precio general del plan. Este precio puede a su vez componerse de un cobro inicial, un cobro periódico y cobros por uso. Puede definirse también una penalización económica por el incumplimiento de la calidad ofrecida. Esta penalización es dependiente del nivel del incumplimiento.

Los contratos entre productos e individuos se representan también mediante un lenguaje que referencia el producto contratado, el plan elegido dentro del modelado de precio y las características seleccionadas si las hubiera. También permite indicar un límite de gasto periódico impuesto por el usuario.

Todos estos elementos deben ser siempre coherentes entre sí, no permitiéndose inconsistencias entre productos y modelos de precio o contratos.

2.2. Contratación y cancelación

El sistema permite a los usuarios contratar productos mediante la selección de un modelo de precio y unas opciones sobre las características si las hubiera. Este contrato tiene una validez determinada dependiente del plan de precio.

Los usuarios también pueden cancelar los contratos que han establecido. En este caso, se aplicará una penalización económica si así se ha definido en el plan de precio.

2.3. Creación y borrado de productos

El sistema permite la creación de productos mediante la definición de un producto y un modelo de precio asociado en el lenguaje de modelado definido.

Además, también es posible borrar un producto definido. En este caso, el producto no se elimina del sistema sino que se marca como producto no válido para la contratación. Los contratos definidos sobre dicho producto, si los hubiera, se cancelarán sin penalización económica alguna.

2.4. Obtención de los datos de uso

Los datos de uso se generan de forma externa al tarificador. Sin embargo, el sistema tarificador proporciona funcionalidad para la escritura de información de uso correspondiente a datos de usos puntuales, continuos o incumplimientos del nivel de servicio.

2.5. Facturación

El sistema genera una factura para cada contrato vigente con la frecuencia establecida en el plan de precio elegido en el mismo. Esta factura se genera de forma automática sin intervención humana y queda almacenada para su posterior consulta. Para su generación, se tiene en cuenta el contenido del plan de precios elegido durante la contratación y el uso asociado a dicho contrato en el periodo a facturar.

Además, se genera una factura inmediatamente después de la contratación para facturar el precio inicial del contrato. También se genera una factura automática tras el borrado de un producto o la cancelación de un contrato.

El sistema también permite la generación de facturas manuales. Estas facturas sirven únicamente como consulta del gasto parcial realizado durante un periodo. Por tanto, no se almacenan de forma posterior a su consulta.

Toda la funcionalidad que se ha descrito para la aplicación está representada en el diagrama de casos de uso que se muestra en la figura 2.1.

2.6. Interfaz de usuario

Además, el sistema define e implementa una interfaz de usuario que permite a proveedores de servicios y clientes de los mismos interactuar con el tarificador por medio de la definición de productos y modelos de precio, el borrado de los mismos, la contratación de productos y su cancelación y la generación y obtención de facturas. Para la definición de productos, modelos de precio y contratos, esta interfaz ofrece una abstracción que no requiere de la edición de documentos en el lenguaje de modelado definido y permite el uso a usuarios menos avanzados de forma intuitiva y usable.

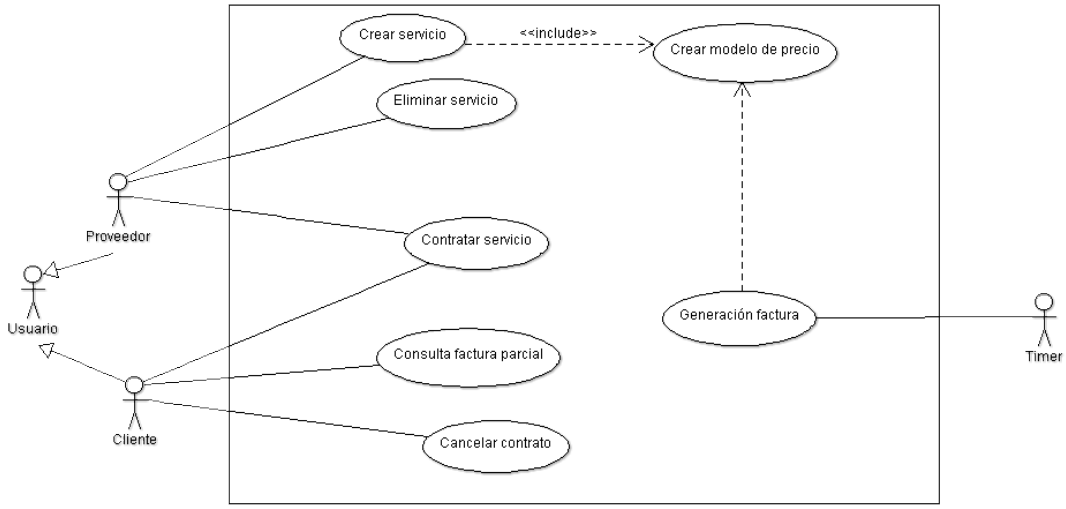

Figura 2.1: Diagrama de casos de uso de la aplicación

3

Estado del arte

3.1. Introducción

Tradicionalmente, en los intercambios comerciales convencionales, los precios han estado fijados desde el inicio, independientemente del cliente, el volumen de compra o el momento de la misma. Estos intercambios han involucrado la fabricación de un producto con un coste asociado que se ha tratado de amortizar [1].

En los últimos años ha proliferado la creación de software como producto comercial. La venta de este tipo de productos debe enfocarse de una forma distinta, puesto que el coste de fabricación no depende del número de unidades vendidas, al ser éstas intangibles.

En este caso, ante una diversidad de clientes y un coste de fabricación fijo, la fijación de precios es una estrategia ineficaz [2]. En su lugar, una asignación estratégica es más beneficiosa.

Más allá de los productos software, Internet ha permitido comerciar con servicios software donde, además de la adquisición, se factura el uso. Estos servicios software pueden adaptarse a las necesidades de un cliente sin que esto incurra en un mayor coste de fabricación y deben ser facturados de forma distinta para cada tipo de uso. Por tanto, su tarificación debe ser planificada, implementada y gestionada con técnicas más novedosas y diferentes a las usadas para los intercambios comerciales tradicionales.

Este Trabajo de Fin de Grado tiene como objetivo el diseño e implementación de un sistema de tarificación de Cloud Services. Para modelar la tarificación de dichos servicios de la mejor

forma posible, se van a estudiar los distintos tipos de precio variable existentes en el mercado, especialmente aquellos aplicables a servicios software. Asimismo, se estudiarán también aquellos sistemas o modelos ya existentes que modelan precios con el objetivo de ver qué funcionalidad y limitaciones ofrecen así como qué tipos de precio permiten modelar para poder establecer comparaciones con el modelo objeto de este trabajo y comprobar que el modelo aquí presentado supera muchas de las limitaciones existentes.

3.2. Modelos económicos de precio

Los modelos de precio variables pueden tomar en consideración distintos parámetros respecto a los cuales realizar esta variación. Nagle y Holden [1] definen las siguientes categorías:

Fijación de precios según el producto

Un mismo producto puede tener asociadas distintas versiones del mismo con distintas características de calidad que permitan asignarle distintos precios según su valor. Asimismo, un producto puede constar de varios precios asociados a varios servicios que oferta para el cliente. De esta forma puede facturarse un coste por contratación y otro por uso o varios costes por el uso de distintos servicios.

Fijación de precios según contadores

Este modelo de precio consiste en el establecimiento de contadores de uso y la facturación según el uso de un servicio. Además, este uso puede tarificarse por segmentos, asociando un valor por uso distinto a cada uno de ellos.

Fijación de precios por segmentos de mercado

Esta estrategia de fijación de precios tiene en consideración las características de cada individuo tales como la edad, el empleo, la capacidad adquisitiva o el lugar de venta para ofrecer un precio distinto por un mismo servicio o producto. Algunos de estos criterios se han llevado a cabo tradicionalmente mediante la fijación de distintos precios en distintos lugares de venta. La venta de servicios software a través de internet no permite esta diferenciación, aunque sí permite realizar una diferenciación por características personales de cada individuo.

Periodos de prueba y reembolsos

Estas dos técnicas de fidelización de clientes consisten en el establecimiento de un periodo inicial gratuito o a menor coste del habitual y de la garantía de devolución de precio en caso de no cumplirse un compromiso establecido por parte del proveedor respectivamente.

Otros modelos de precio

Otros modelos de precio comunes en el mercado diario de servicios son los siguientes:

- Uso ilimitado. Como las tarifas planas de telefonía.
- Pago por consumo. Como el pago por servicio de envío de SMS.
- Pago por tiempo de uso. Como el pago por duración de una llamada.
- Servicio gratuito con extensiones de pago. Como aplicaciones móviles con publicidad que cobran un precio extra por eliminarla.
- Pago por características. Como el de la televisión por cable que ofrece un paquete básico de canales y canales extra a elegir por un suplemento.
- Pago por niveles. Como el servicio Dropbox [3] que consta de un nivel reducido gratuito y varios niveles de pago con distintas características.

Todos ellos son válidos para la tarificación de servicios software, puesto que se trata de funciones del uso según parámetros del mismo.

3.3. Modelos implementales y aplicaciones

Actualmente, la mayoría de los sistemas comerciales de facturación, están enfocados a la venta de productos tangibles, como los sistemas de Terminal de Punto de Venta. En cambio, los sistemas enfocados total o parcialmente a la facturación de servicios que pueden ser usados por un tercero son limitados. Dado que los servicios que se tarifican son informáticos, muchas empresas o proveedores poseen sus propios tarificadores.
A continuación se detallan algunas iniciativas de modelos, diseñados o implementados, de sistemas de tarificación de servicios accesibles por terceros. Se trata, tanto de modelos teóricos, resultado de investigaciones, como de sistemas comerciales.

3.3.1. Modelos

USDL

USDL (Unified Service Description Language) es un lenguaje XML de definición de servicios técnicos y de negocio para su consumo. Contiene un módulo de precios que permite definir un precio variable para un producto o servicio en formato XML. El objetivo de este módulo de precios es modelar las reglas de segmentación de una estructura de precio (las reglas que determinan cómo y cuándo a distintos clientes se les cobran precios distintos) [4]. En [5] se especifican las características generales de este lenguaje en pseudocódigo XML. El lenguaje permite definir distintos planes de precio para un mismo servicio o producto. Para cada uno de estos planes de precio se especifica una moneda y opcionalmente un precio mínimo y un precio máximo de gasto. También es posible asignar uno o varios impuestos. El modelo especifica el inicio de validez de un precio y opcionalmente el fin de la misma.

Cada plan de precio se compone de uno o varios componentes que contribuyen al precio total si se cumplen unas ciertas condiciones. Cada componente consta de uno o varios niveles de precio y unos parámetros o funciones por los que facturar. Los niveles de precio, que son absolutos o proporcionales, pueden a su vez ser negociables por el cliente. La forma y el instrumento de pago también quedan modelados mediante este lenguaje.

Este lenguaje es completo aunque complejo de comprender y manejar. No existe una definición definitiva y estandarizada del mismo ni herramientas de edición o comprobación de ficheros de este lenguaje de acceso público.

Tarificación flexible de servicios de Internet

Tarificación flexible de servicios de Internet [6] es una propuesta teórica de un modelo arquitectónico de tarificación de servicios heterogéneos. Se propone un posible estándar flexible adaptable a cualquier servicio de Internet. Dicho estándar se compone de un método de medición y uno de tarificación y facturación.

Para el almacenamiento de la información de medición de consumo de servicios de Internet se propone el uso de IPDR (Internet Protocol Detail Record) [7]. Este protocolo, definido específicamente para la tarificación de servicios heterogéneos cuenta con un sistema de almacenamiento de información y otro de transmisión de la misma.

Para la tarificación y facturación se propone el uso del sistema JBilling [8]. Dicho sistema se define como ágil y flexible y permite modelar la tarificación según reglas de negocio. Permite distintos periodos de facturación y la definición de promociones y agregados.

Esta definición teórica toma dos herramientas ya existentes y define un sistema completo de tarificación de altas prestaciones. No existe, en cambio, una implementación de dicha definición que integre ambas herramientas.

3.3.2. Aplicaciones

JBilling

JBilling [8] es un software Open Source de facturación. Se trata de un software de pago para la facturación en empresas. Consta de varias versiones con distintas funcionalidades. La versión gratuita no cuenta con ninguna de la funcionalidad avanzada de modelado de precios. Cuenta con una interfaz de usuario de uso del sistema de tarificación que permite editar los precios y los productos, leer ficheros de uso de servicios en formato CDR [9] y obtener facturas en formato PDF. Cuenta además con procesamiento de pago automatizado mediante tarjeta de crédito.

El modelado de precios permite crear promociones y conjuntos de productos. Es posible crear varios planes y modelos de precio que definan un precio compuesto para un producto.

Este producto es flexible y está orientado a la tarificación de servicios. Permite crear modelos de precio complejos basados en reglas. Aun así, la funcionalidad de modelado de precios es únicamente de pago.

FreeSide

FreeSide [10] es un sistema Open Source de monitorización y tarificación de servicios de red. Cuenta con un sistema de monitorización en tiempo real, un sistema de facturación y un sistema de cobro automático a través de Internet. Permite crear modelos de precio flexibles mediante la creación de modelos de precio, ofertas y segmentación de precio.

Se trata de un sistema completo de monitorización y tarificación orientado a proveedores de servicios. No permite la tarificación general de cualquier servicio con datos de uso externos y su manejo es altamente complejo.

3.4. Conclusiones

A pesar del limitado número de modelos de precio existentes, se han detectado varios de ellos comúnmente usados para la facturación de servicios. Es necesario un sistema que permita representar todos estos modelos para cualquier servicio. Se han visto sistemas genéricos, pero de uso limitado y arquitectura interna desconocida. Se han encontrado modelos genéricos que no han sido estandarizados y se han encontrado propuestas de arquitectura no implementadas.

Como se ha visto, se trata de un área poco explorada y con escasas implementaciones. Estas implementaciones, además, tienen importantes carencias tal y como se han ido exponiendo en los apartados anteriores. Por tanto, este estudio muestra que sería de importante relevancia y utilidad la creación de un sistema de tarificación para cualquier servicio que abarque todos los posibles modelos de precio detectados y sea de uso sencillo para un proveedor.
Esto es lo que propone el presente trabajo, proporcionando además un importante avance en el sector de tarificación de servicios de Internet.

4

Definición del proyecto

4.1. Objetivos

Este proyecto tiene como objetivo la definición, implementación y validación de un sistema electrónico de facturación de Cloud Servicies (SaaS, PaaS, CaaS o NaaS) ofrecidos por un proveedor de telecomunicaciones mediante la definición de un modelo genérico de modelado de precios y el diseño de un algoritmo de facturación sobre el mismo.

Este trabajo está enmarcado dentro del proyecto TESLA, financiado por la Junta de Castilla y León cuyo propósito es la implementación de una plataforma global de distribución de servicios de telecomunicaciones basados en la red de Telefónica. Además, ofrece capacidades como el control de llamadas, la mensajería, la localización o la comunicación M2M. La validación del sistema resultante de este trabajo se realizará en el entorno de pruebas de dicho proyecto tarificando los servicios correspondientes a las capacidades ofrecidas por el mismo.

Todo el proceso de desarrollo de este trabajo ha sido realizado íntegramente por la estudiante, incluyendo el análisis del problema, el diseño de soluciones y la implementación completa de las mismas. Este desarrollo se ciñe únicamente a la necesidad de un tarificador de servicios dentro del proyecto TESLA y a la integración con el resto de elementos del mismo.

Los usuarios potenciales de este sistema son los proveedores de servicios de internet y los clientes que contraten dichos servicios.

Los usuarios proveedores pueden definir las características y parámetros del servicio que desean ofertar y crear un modelo de precio para el mismo. Pueden asimismo asignar un precio a cada tipo de interacción según el momento de uso, el volumen de uso y el uso acumulado. Cada modelo de precio puede estar asignado a una o más divisas o países.

Por otra parte, los usuarios clientes de servicios pueden contratar un servicio existente por un periodo de tiempo, obtener la correspondiente facturación del mismo de forma periódica según los datos de uso y cancelar el contrato antes de su finalización obteniendo un cálculo de la penalización correspondiente según el modelo de precio. Además, pueden obtener una factura parcial antes de la finalización del periodo con el coste actual de la misma.

Además la herramienta permite asignar varios modelos de precio a un mismo servicio y ofrecer distintos niveles de calidad del mismo con distintos modelos de precio.

El sistema también gestiona la interacción con clientes y proveedores y toma los datos de uso de los servicios por parte de los clientes de una fuente externa. Con ellos calcula la factura de cada cliente. Asimismo, se ha desarrollado una interfaz de usuario intuitiva y usable para gestionar las funcionalidades y características antes mencionadas.

Mediante esta herramienta los distintos usuarios pueden tanto ofertar como contratar servicios de internet de forma dinámica e intuitiva asegurando la calidad del proceso.

4.2. Acotación

El sistema permite a los proveedores definir productos y modelos de precio así como a los clientes contratar dichos productos. El uso de estos productos y la generación de datos de uso para la facturación no son gestionados por esta aplicación, que sí los usa para generar las correspondientes facturas.

Esta aplicación tampoco interactúa con sistemas de pago para la realización del cobro efectivo del monto indicado en cada factura.

4.3. Definición de conceptos

Cliente Usuario del sistema cuyo rol es la contratación y el uso de servicios o productos.

Proveedor Usuario del sistema cuyo rol es la oferta de servicios o productos para su consumo por parte de clientes.

Producto Conjunto de funcionalidad ofrecida por un Proveedor a un Cliente que puede contratarse en una transacción. Podrá ser referido también por el término Servicio.

Componente Cada una de las funcionalidades independientes que un Producto ofrece a un Cliente. En un producto monoproceso, cada vez que el cliente interactúa con el mismo hace uso de un único componente.

Unidad de uso Mínima medida de un Componente por cuyo uso se puede cobrar a un Cliente. Un Componente puede contar con una o varias medidas.

Parámetro Característica de un componente con valor variable por el proveedor. Durante la contratación, si el proveedor así lo ofrece, el cliente puede seleccionar un valor del parámetro para su uso del producto.

Precio base Precio compuesto por un precio inicial único, un precio periódico y un periodo definido para dicho precio (en días, semanas, meses o años).

Plan Conjunto de precios, tipos de moneda y opciones sobre parámetros asociadas a un producto. Definen las monedas para las cuales se encuentra disponible la contratación del mismo así como los precios de facturación del mismo. Durante el proceso de contratación, cada cliente debe escoger un único plan para un producto. El conjunto de precios se compone de un precio base, una duración establecida para los contratos de dicho de producto y una tarificación para cada componente del producto. Cada tarificación por componente es un conjunto de precios por unidad de uso según distintas condiciones temporales o de uso previo. Para cada parámetro variable de un componente se define una tarificación que puede además incluir un precio base. Puede opcionalmente incluir un máximo o mínimo para el precio periódico del producto.

Modelo de precio Conjunto de planes ofertados para un producto. Cada producto está asociado a un único modelo de precio. Durante la contratación de un producto el cliente debe elegir un único plan.

Contrato Acuerdo entre el proveedor y el cliente para el uso de un producto por parte del segundo. Cada contrato especifica un plan para un producto, el valor seleccionado para cada parámetro si hubiese varios y la duración del mismo.

4.4. Metodología utilizada

El desarrollo de este sistema ha seguido un ciclo de vida en cascada con realimentación. Se ha elegido este ciclo de vida ya que, dado el tamaño del sistema, ha permitido la realización del mismo en un tiempo menor que si se realizasen varios incrementos y ha permitido realizar una mejor estimación del trabajo a realizar y el tiempo que éste ha conllevado que el uso de metodologías ágiles. La realimentación ha permitido realizar modificaciones necesarias en alguno de los aspectos de fases ya concluidas con el fin de perfeccionar el sistema.

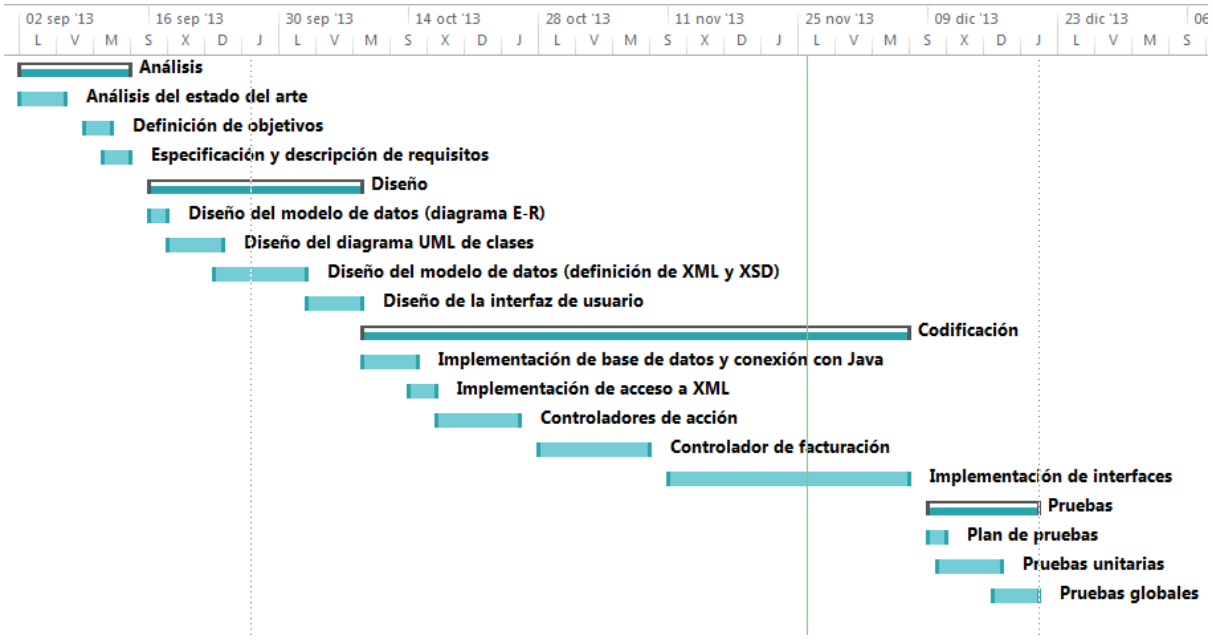

Figura 4.1: Diagrama Gantt con la planificación temporal del proyecto.

A continuación se detallan las fases que se han llevado a cabo durante el desarrollo, las técnicas y herramientas utilizadas en cada una de ellas y las entradas y salidas esperadas. En el diagrama Gantt de la figura 4.1 puede observarse la distribución temporal de dichas tareas así como las dependencias entre las mismas.

Análisis

Tareas

- Análisis del estado del arte.
- Definición de objetivos.
- Especificación y descripción de requisitos funcionales y no funcionales.

Salidas esperadas: Definición y acotación del sistema. Listado de requisitos.

Diseño

Tareas

- Diseño de un diagrama E-R para la base de datos.
- Diseño UML del sistema a implementar.
- Diseño de los ficheros XML de información de productos y precios.
- Diseño de la interfaz de usuario.

Entrada esperada: Listado de requisitos.
Salida esperada: Diagramas de diseño.

Codificación

Tareas

- Codificación de la base funcional de la aplicación.
- Codificación de las reglas de precios de la aplicación.
- Codificación de la interfaz.

Entrada esperada: Información de diseño.
Salida esperada: Código de la aplicación.

Pruebas y validación

Tareas

- Realización de un plan de pruebas.
- Realización de pruebas unitarias.
- Realización de pruebas globales de caja negra de integración y de sistema.
- Evaluación del sistema final.

Entrada esperada: Código de la aplicación.
Salida esperada: Código de la aplicación validado y verificado.

4.5. Herramientas utilizadas

Para la implementación de este sistema han sido necesarias las siguientes herramientas:

- Un lenguaje de definición y manejo de bases de datos.
- Un lenguaje de programación orientado a objetos.
- Un lenguaje de definición de información textual.
- Una estructura de manejo de interfaces vista-controlador.
- Una API para la generación de ficheros PDF.

A continuación se detallan las herramientas que han sido utilizadas para cada propósito, haciendo énfasis en aquellas menos conocidas o extendidas.

Definición y manejo de bases de datos: MySQL e Hibernate

Para la definición de la base de datos se ha utilizado el lenguaje SQL, implementando dicha base de datos en el sistema de gestión MySQL[11]. A diferencia del sistema PostgreSQL utilizado

en la universidad, el uso de este gestor de código abierto está más extendido. También ha sido necesario realizar una conexión de dicha base de datos con el código de la aplicación. Esto se ha realizado a través de una conexión Hibernate.

Hibernate[12] es una herramienta de conexión de bases de datos relacionales con Java mediante el uso de etiquetas o ficheros . Esta conexión permite acceder a la base de datos tratando sus tablas como Beans. Se trata de una herramienta portable entre distintas bases de datos que permite abstraer el lenguaje específico de cada una de ellas usando en su lugar HQL (Hibernate Query Language)[13] para acceder a cada una de ellas. Esta portabilidad hace que Hibernate sea un sistema de conexión más robusto que el sistema JDBC[14] estudiado en la universidad y que, por tanto, su uso esté más generalizado.

Lenguaje de programación: Java

El código base de esta aplicación se ha implementado en lenguaje Java ya que se trata de un lenguaje orientado a objetos altamente extendido y portable entre distintos sistemas. Además, se trata de un lenguaje con soporte para programación web y gran número de APIs y herramientas específicas como las que se detallan en los apartados siguientes.

Definición de información textual: JAXB

Ha sido necesario un lenguaje de definición de productos, modelos de precio y contratos que abstrajese estos de la base de datos pudiendo ser almacenados en ésta como una única unidad. Para esta tarea se ha elegido el lenguaje XML (Extensible Markup Language)[15], que permite almacenar información en una estructura de árbol muy útil para el propósito de esta aplicación. Se trata de un formato legible por un ser humano a diferencia del almacenamiento en base de datos. Esto permite la edición manual de los ficheros representados en esta estructura por parte de un usuario avanzado. Además, es posible definir mediante XSD (XML Schemas)[16] un formato específico para cada fichero XML y comprobar que satisface este formato. Los ficheros en lenguaje XML pueden también ser accedidos con facilidad desde código Java mediante herramientas como JAXB[17]. Esta herramienta permite representar los ficheros XML como clases Java -creadas automáticamente a partir de su esquema de definición XSD[16]- para su lectura y edición.

JAXB[17] es una API que permite transformar representaciones XML en clases Java a partir de sus esquemas de definición XSD[16]. Además, permite almacenar objetos Java como XML y transformar XML en objetos Java abstrayendo la complejidad del manejo de XML desde Java. Se ha seleccionado JAXB[17] para el desarrollo de esta aplicación dada su simplicidad en el acceso de ficheros XML. El acceso a ficheros XML y la creación o edición de los mismos es frecuente en esta aplicación, por lo que la agilización de este proceso es beneficiosa para la misma.

Aunque la edición de ficheros XML sí ha sido tratada en la universidad, la creación de ficheros XSD de definición es una tarea novedosa para la estudiante. Asimismo, el manejo de los ficheros XML desde código Java mediante JAXB ha sido aprendido para la realización de este trabajo.

Estructura de manejo de interfaces vista-controlador: Spring

La interfaz de la aplicación se ha estructurado según el modelo vista-controlador que separa la interfaz visual de la funcionalidad que presenta la misma y facilita la gestión posterior de su código. Para ello se ha usado el módulo de modelo vista-controlador del Framework Spring[18], basado en HTTP y Servlets. Spring[18] es un framework de desarrollo para Java con múltiple funcionalidad, entre la que se encuentra la antes mencionada. Sus ventajas radican en la modularización, con un bajo acoplamiento entre la interfaz y la funcionalidad que ésta ofrece y gran facilidad para la realización de pruebas. El Framework Spring[18] no había sido utilizado previamente en el ámbito universitario y su aprendizaje ha sido necesario para la realización de este proyecto.

La vista de las interfaces se implementará en JSP (Java Server Pages)[19]. Además, se ha utilizado Javascript para la dinamización de estas páginas durante su uso. Las tecnologías JSP y Javascript ya habían sido usadas durante otros proyectos en el ámbito universitario.

API para la creación de ficheros PDF: JasperReports

Para una mejor visualización de las facturas resultantes de la tarificación, este sistema genera una versión PDF de las mismas. Para la generación de dichos ficheros PDF se ha hecho uso de la herramienta JasperReports[20] compatible con Java. Esta herramienta permite generar informes en distintos formatos con el objetivo de que sean cómodamente legibles por el usuario.

La API JasperReports[20] no había sido utilizada con anterioridad, y se ha aprendido su uso para este proyecto.

5 Requisitos

5.1. Introducción

A continuación se enumeran los requisitos de la aplicación que se va a diseñar e implementar. Estos requisitos son el resultado del análisis del problema propuesto: diseñar un tarificador de servicios de internet. Este análisis se ha realizado mediante la investigación del estado del arte y la evaluación de las necesidades generales de los usuarios de la tarificación de servicios y de los requerimientos específicos del proyecto TESLA.

5.2. Requisitos funcionales

RF. 1 *Los proveedores dados de alta en el sistema podrán definir servicios identificando los componentes y parámetros de los mismos. Para cada parámetro, se especificarán los posibles valores, continuos o discretos, que puede tomar.*

RF. 2 *Cada servicio podrá tener un número ilimitado de componentes, con un mínimo de uno.*

RF. 3 *Cada componente podrá tener un número ilimitado de parámetros, pudiendo no tener ninguno.*

RF. 4 *La definición de cada servicio incluirá las unidades de medida del uso de dicho servicio.*

Todo servicio tendrá definida al menos una unidad de medida. El número de unidades será ilimitado.

RF. 5 *Los proveedores tendrán que asignar un modelo de precio a cada servicio definido. Cada modelo de precio se compondrá como mínimo de un plan , que tendrá definido un precio base y opcionalmente precios por uso de los componentes.*

RF. 6 *Cada plan dentro de un modelo de precio podrá asociarse a una o varias divisas o países. De esta forma, dicho plan sólo se encontrará disponible para aquellos clientes que pertenezcan a dichos países o hagan uso de dichas divisas.*

RF. 7 *Los precios por uso de los componentes dependerán de las unidades de medida especificadas en la definición del servicio.*

RF. 8 *En cada modelo de precio se especificará la duración mínima de contratación del servicio asociado. Opcionalmente se podrá especificar el coste de cancelación temprana para el cliente, que podrá depender del tiempo restante de contrato.*

RF. 9 *Se especificarán en el modelo de precios aquellos parámetros con valores variables identificándose los valores admitidos y el precio por la selección de cada uno de ellos. Asimismo, se especificará la variación de dicho precio en caso de incumplimiento del acuerdo de servicio.*

RF. 10 *Cada modelo de precio establecerá un periodo de cobro para el servicio asociado que podrá ser de días, semanas, meses o años.*

RF. 11 *Cada modelo de precio podrá tener asociado un precio mínimo y/o máximo independiente del uso del mismo.*

RF. 12 *El sistema permitirá a los proveedores eliminar servicios, suspendiéndose los contratos asociados a los mismos y no realizándose cobros correspondientes a uso posterior de dichos servicios.*

RF. 13 *Cada cliente podrá contratar un servicio de los publicados por los proveedores. Tendrá que especificar el plan de precio elegido de entre los ofertados en el modelo de precio y el valor elegido para cada parámetro variable.*

RF. 14 *Cada cliente podrá cancelar los contratos que ha realizado previamente. La cancelación tendrá como consecuencia el pago especificado en el modelo de precio del servicio contratado.*

RF. 15 *El sistema generará automáticamente una factura cuando se produzca el vencimiento del periodo asociado a cada contrato. Dicha factura tomará datos de uso de procedencia externa para calcular el total. Cada factura mostrará un desglose de los cargos que la componen especificando el uso realizado sobre un componente si lo hubiera.*

RF. 16 *El sistema permitirá a los clientes establecer un límite de gasto para cada contrato en cada periodo. Al alcanzarse dicho límite el proveedor no prestará más servicio hasta el final del periodo.*

RF. 17 *Dado un evento de uso que corresponda a varias caracterizaciones en el modelo de precios correspondiente, siempre se facturará con la que resulte en un coste menor.*

RF. 18 *El sistema permitirá generar facturas intermedias manuales con la información de uso de un periodo no finalizado. Esta información de uso seguirá disponible para la facturación automática al final del periodo.*

5.3. Requisitos no funcionales

RNF. 1 *Los proveedores podrán definir los servicios y sus modelos de precio a través de una interfaz de usuario interactiva e intuitiva.*

RNF. 2 *La interfaz de definición de servicios y precios no permitirá a los usuarios generar inconsistencias en dichos modelos.*

RNF. 3 *Los clientes podrán contratar servicios y cancelar dicha contratación a través de una interfaz gráfica de usuario. Además, podrán obtener el resultado de las facturaciones automáticas periódicas sobre dicho contrato así como generar una facturación manual en cualquier instante.*

RNF. 4 *Los datos de uso de un cliente se procesarán automáticamente una única vez.*

6

Diseño del modelo

6.1. Introducción: arquitectura general

Para cumplir con los requisitos especificados en el capítulo 5 y las funcionalidades definidas, la aplicación se estructura en los siguientes componentes básicos, que se muestran en la figura 6.1

- Una base de datos relacional en la que almacenar la información de usuarios, la definición de productos, los modelos de precio, los contratos realizados y la información de uso asociada a cada uno de ellos. La información específica relativa a productos, precios y contratos se almacena en ficheros XML[15] con formato específico. La información relativa al uso de los productos contratados se almacena dentro de la base de datos en JSON[21] con formato específico.

- Un gestor de acciones que incluye la creación y borrado de productos y modelos de precio y la creación y cancelación de contratos. Este gestor conecta el código de la aplicación con la base de datos. Además, comprueba la coherencia entre los distintos XML de definición cada vez que se crean, editan o borran los mismos.

- Un gestor de facturación que gestiona la transformación de los datos de uso en una factura. Esta transformación se realiza con un particionamiento y un posterior cálculo del precio. La salida resultante se presenta como un XML, que es a su vez transformado en una factura en PDF para una mejor visualización. El gestor de facturación se activa automáticamente

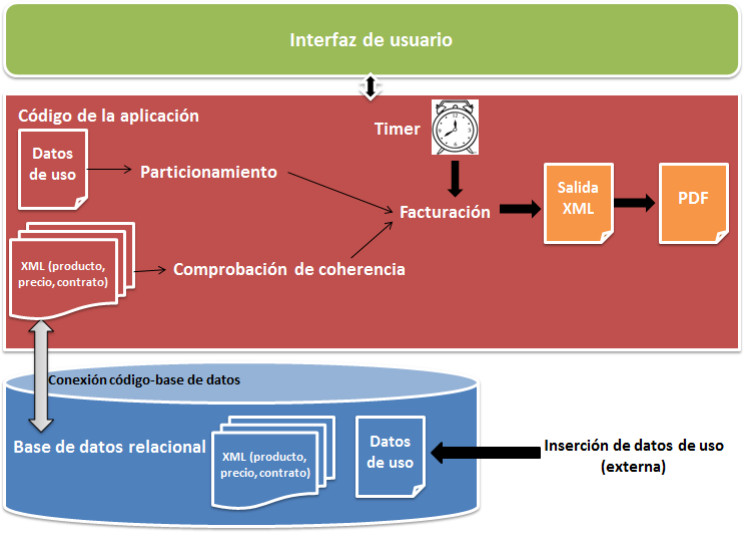

Figura 6.1: Diagrama de la arquitectura general de la aplicación.

cuando venza un periodo de facturación para un contrato. Para ello, un timer se encarga de realizar dicha comprobación.

- La interfaz de usuario que, haciendo uso del resto de componentes, permite definir y borrar productos y modelos de precio, así como realizar contratos y cancelar los mismos. También permite ejecutar una facturación manual de forma intuitiva y visualizar el resultado de la facturación (manual o automática).

6.2. Base de datos

El sistema se ha construido sobre una base de datos relacional que almacena toda la información de productos, modelos de precio, datos de uso y contratos que se utilizan en el mismo. En la figura 6.2 se muestra el diagrama entidad-relación de dicha base de datos. Además, a continuación se explican las entidades y relaciones contenidas en el mismo.

Entidades

El modelo consta de las siguientes entidades, que representan cada uno de los elementos conceptuales que maneja la aplicación.

- **EM_SP_PRODUCT** representa un **producto** manejado por la aplicación, representado mediante los siguientes campos:
  - nu_productid, **identificador** numérico del producto.
  - tx_productname, **nombre** del producto.

  - tx_productdesc, **descripción** del producto.
  - b_definition, **XML de definición** del producto.
  - dt_valid_from, **fecha de creación** del producto.
  - dt_valid_to, **fecha de finalización** de la validez (en caso de borrado por parte del proveedor).

- **EM_PARTY** es la entidad encargada de representar una abstracción de los **usuarios** de la aplicación. Estos usuarios pueden ser tanto clientes como proveedores. La información específica de clientes y proveedores se representa en dos tablas denominadas EM_INDIVIDUAL y EM_ORGANIZATION que no son accedidas por el tarificador, pero a las que referencia esta entidad. Los usuarios se representan mediante la siguiente información:

  - nu_party_id, **identificador** numérico de la entidad.
  - tx_party_identity_type, **tipo** de la entidad (individuo u organización).
  - nu_individual_id, **identificador del individuo** en el caso de un cliente.
  - nu_organization_id, **identificador de la organización** en el caso de un proveedor.

- **EM_PRICE_MODEL** representa el **modelo de precio** asociado a un producto. Dicho modelo define las opciones disponibles para la contratación de dicho producto y cómo se realiza la facturación para cada una de ellas. Cuenta con la siguiente información:

  - b_price, **XML de definición** del modelo de precio.

- **EM_CONTRACT** representa un **contrato** realizado entre un cliente y un producto ofertado. El contrato especifica las opciones seleccionadas dentro del modelo de precio, así como información de despliegue de la aplicación para dicho cliente.

  - nu_provision_id, **identificador** numérico del contrato.
  - tx_deployment_id, identificador de **despliegue** del producto.
  - b_contract, **XML de descripción** del contrato.
  - dt_date_begin, **fecha de inicio** del contrato.
  - dt_date_end, **fecha de finalización** del contrato.
  - dt_next_billing, **fecha de próxima facturación** del contrato.

- **EM_USAGE_DATA** representa un **dato de uso** de un usuario a una aplicación. Es por esto que se asocia a un contrato. Dicho dato de uso tiene asociada la fecha de ocurrencia del mismo así como un indicador que indica si ya ha sido procesado y evita procesarlo dos veces. Los datos que lo representan son los siguientes:

  - b_data, **información de uso** en formato SDR.

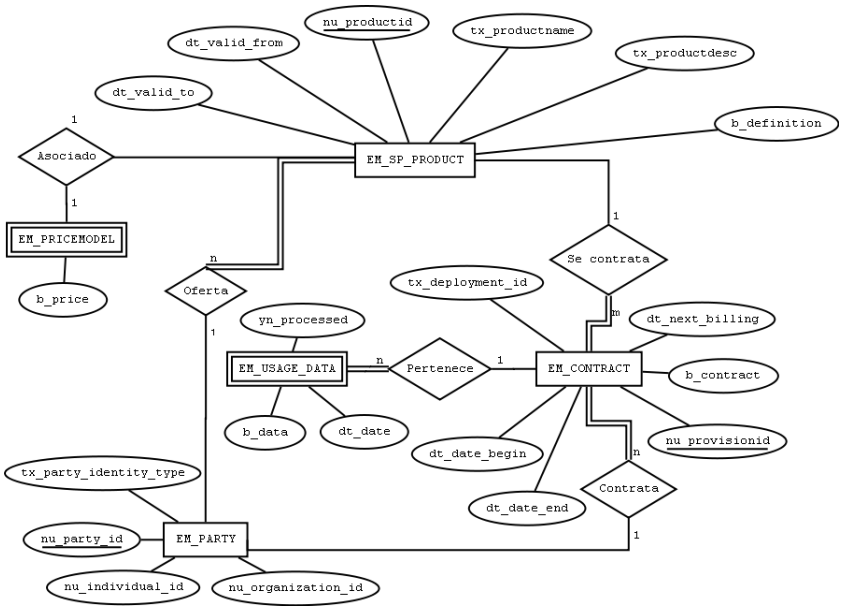

Figura 6.2: Diagrama entidad-relación de la aplicación.

- dt_date, **fecha del uso**.
- yn_processed, **indicador** de si el dato ha sido procesado.

Relaciones

El modelo consta de las siguientes relaciones entre entidades del mismo que coinciden con las relaciones conceptuales entre los mismos:

- Relación de un producto con el proveedor que lo oferta para la identificación del mismo.
- Asociación de un dato de uso a un contrato. Esta asociación permite identificar los datos de uso a facturar al producirse el vencimiento de un periodo para un contrato dado.
- Asociación de un contrato al cliente que lo realiza.
- Asociación de un contrato con el producto que se contrata.
- Asociación de un modelo de precio al producto cuyo precio modela.

6.3. Definición de datos en XML

La información detallada de los productos, los modelos de precio asociados a los mismos y los contratos realizados sobre un producto se representan mediante XML[15]. Estos XML tienen un formato específico para cada tipo y se encuentran almacenados en la base de datos. El formato de cada uno de estos XML está definido mediante un fichero con formato XSD[16] cuyo contenido completo se muestra en el Anexo A.

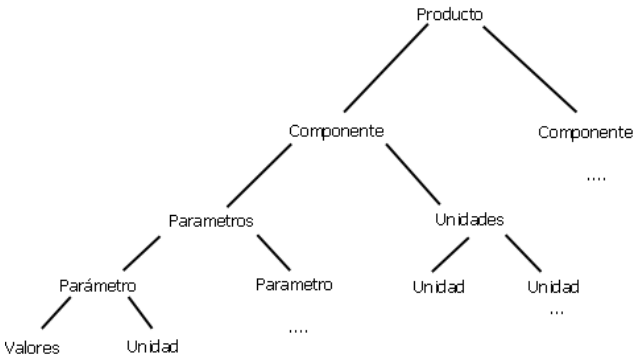

Figura 6.3: Esquema de la información del XML de definición de producto.

6.3.1. Definición de producto en XML

En la figura 6.3 se muestra un esquema en forma de árbol de la organización del XML de definición de producto. En él se observa que para cada producto es posible definir uno o más componentes. Cada uno de estos componentes representa una funcionalidad distinta ofrecida a los clientes.

Para cada componente es posible definir un conjunto de parámetros. Cada parámetro define una característica que es posible ofrecer a un cliente con distintos valores a seleccionar durante la contratación.

Para cada parámetro se define un conjunto o rango de valores posibles así como la unidad en la que éstos se miden. Estos valores podrán ser referenciados desde el modelo de precio para asociar un precio a la selección de cada uno de ellos.

Además, cada componente tiene definidas unas unidades según las que se tarificará (por ejemplo invocación, minuto, segundo, llamada). Cada unidad representa la cantidad mínima de uso que se tarificará. Pueden definirse varias unidades para un mismo componente de forma que cada dato de uso esté medido en uno de ellas.

6.3.2. Definición de modelo de precio en XML

En la figura 6.4 se muestra un esquema en forma de árbol del XML de definición de modelo de precio. Cada modelo de precio se define como un conjunto de planes que podrán contratarse de forma excluyente. Es decir, cada usuario elige un único plan al realizar la contratación para que se le apliquen los precios definidos en el mismo. Cada uno de estos planes tiene establecido un conjunto de monedas y países en los que es válido. De esta forma, los usuarios que no pertenezcan a ninguno de los países indicados o no usen ninguna de las monedas indicadas no podrán contratar dicho plan.

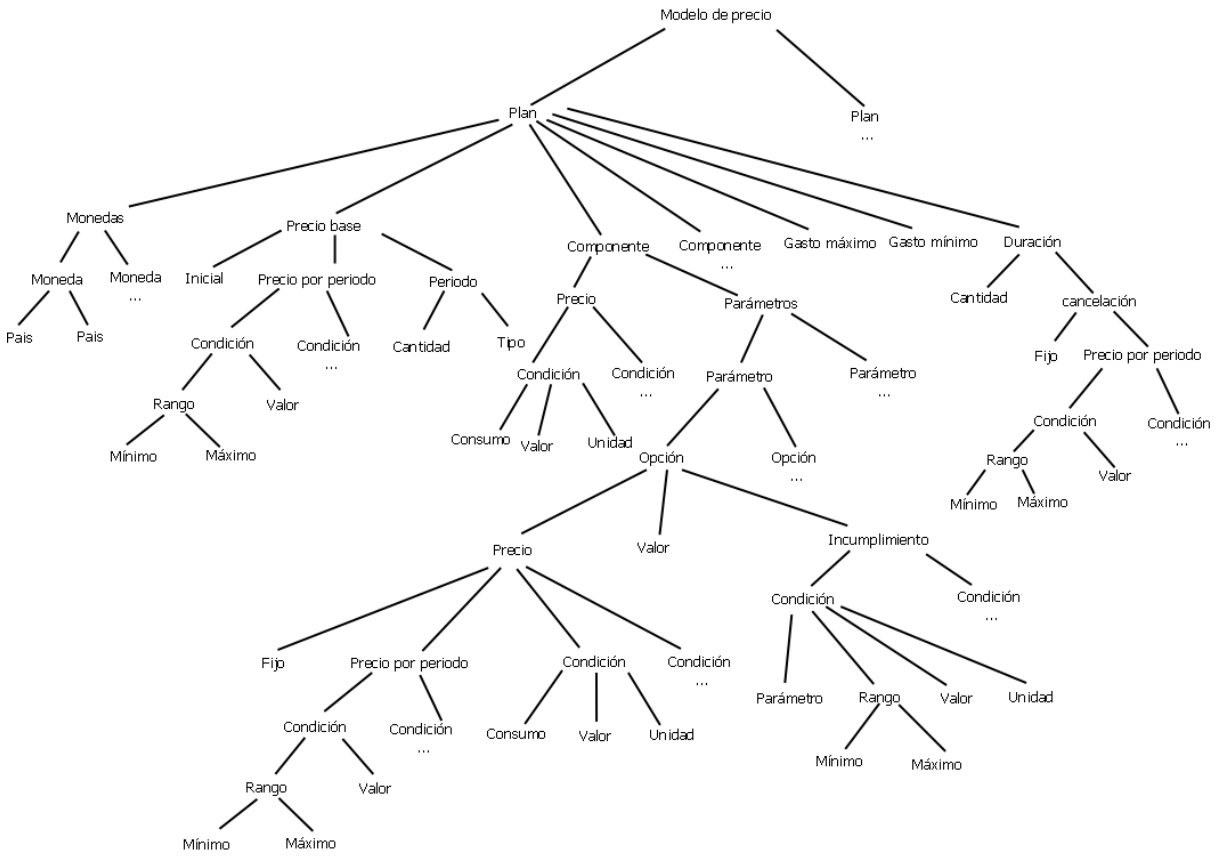

Figura 6.4: Esquema de la información del XML de definición de modelos de precio.

El precio para cada plan se define como un precio base general y un precio por componente. Este precio base se compone a su vez de un precio único de contratación y un precio periódico. El primero de ellos es aquel que se cobrará al cliente únicamente en el momento de contratación mientras que el segundo es aquel que se cobrará cada vez que se realice una facturación periódica.

El precio periódico puede definirse de forma única o tener un valor diferente para cada periodo. De esta forma, si un contrato se factura de forma mensual, es posible especificar que el primer mes tenga un precio periódico de 10 € y el resto de los meses tengan uno de 20€.

El periodo de facturación se representa en función de periodos simples. Estos periodos simples son días, semanas, meses o años. Es necesario especificar qué cantidad de ellos transcurrirá entre las facturaciones.

Cada componente tiene definido un conjunto de precios. Estos precios corresponden a una cantidad a cobrar por unidad de uso. Estas unidades de uso son las definidas para el componente en la definición del producto. Cada uno de los precios puede ser incondicional (aplica a cualquier uso) o aplicable según una condición. Estos son los tipos de condiciones existentes:

1. Uso acumulado actual. Esta condición define un intervalo de uso dentro del cual aplica. Este intervalo debe darse dentro de un mismo dato de uso. Así, esta condición aplica a aquellos usos unitarios donde la cantidad se encuentre en el intervalo especificado en esta condición o a la parte de aquellos usos temporales donde la duración esté entre el mínimo y el máximo del intervalo. Así, una condición puede referirse al uso entre los minutos 5 y 10 y, por tanto, la parte de un uso temporal que se encuentre entre el quinto y el décimo minuto se facturará de este modo.

2. Uso acumulado total. Esta condición también define un intervalo de uso dentro del cual aplica, pero dicho intervalo puede darse como resultado del acumulado de varios usos. Así, esta condición aplica a aquellos usos unitarios que supongan el n-ésimo uso del componente para la unidad dada durante el periodo y tal que n pertenezca al intervalo definido.

3. Periodo de tiempo. Esta condición se refiere a un intervalo definido en días de la semana y/o horas del día. La condición aplica a todos aquellos usos (o partes de los usos en el caso de usos temporales) que se hayan producido en un día de la semana perteneciente al intervalo y/o en una hora perteneciente al intervalo según si están definidos ambos intervalos o sólo uno de ellos.

4. Uso acumulado en un periodo de tiempo. Esta condición es una combinación de las dos anteriores donde se define un intervalo de uso total acumulado, pero que sólo tiene en cuenta el uso realizado durante el periodo definido en forma de intervalos de días de la semana u horas del día.

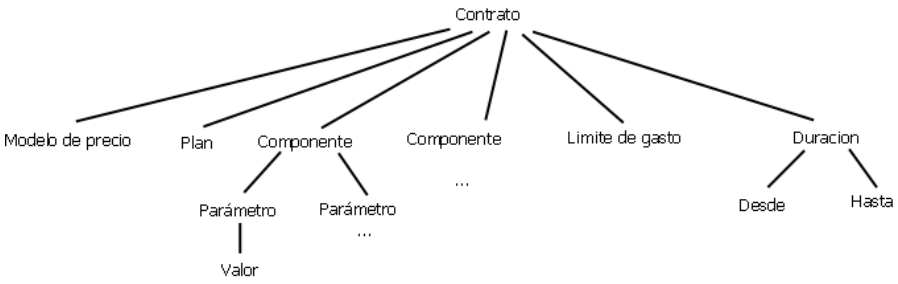

Figura 6.5: Esquema de la información del XML de definición de contrato.

Implementalmente las tres últimas condiciones se definen dentro de la misma estructura (uso-total). Nótese que todos los intervalos definidos en el modelo de precio son cerrados, considerándose contenidos en ellos los valores de los extremos.

Las condiciones definidas pueden solaparse, de forma que un dato de uso pueda verse afectado por varias de ellas. En este caso, durante el proceso de facturación, se aplicará el precio más bajo de entre los posibles.

Además de para los componentes, para cada valor de un parámetro puede definirse un precio como los anteriores, con los mismos tipos de condiciones. Asimismo, puede definirse una penalización por incumplimiento dependiente de la magnitud del mismo. Por otra parte, cada plan de precio puede tener definido un gasto máximo y/o un gasto mínimo. Estos gastos actúan como topes en la facturación periódica, facturándose siempre dentro del intervalo definido entre ambos, independientemente de los datos de uso.

Cada plan de precio tiene también definida una duración del contrato que se realice sobre el mismo. Esta duración puede definir también un coste por cancelación temprana, antes de la finalización del contrato. Este coste puede ser fijo (independiente del momento de la cancelación), dependiente del tiempo restante (con un coste definido para cada mes restante, de forma que se cobre la suma de todos ellos) o una combinación de ambos.

6.3.3. Definición de contrato en XML

En la figura 6.5 se muestra un esquema en forma de árbol del XML de representación de contrato. Se puede observar que el contrato consta únicamente de referencias al modelo de precio indicando las elecciones del usuario durante la contratación. De esta forma, el XML almacena una referencia al plan elegido y a los valores seleccionados para cada parámetro. Además, se almacena la duración temporal representada como intervalo. Esto permite conocer el periodo de validez del contrato. Si el usuario estableciera un límite de gasto, este también se almacenaría en el contrato.

En el anexo C pueden verse ejemplos de ficheros de los tipos anteriormente definidos.

6.4. Gestores

La aplicación cuenta con un gestor de acciones y un gestor de facturación. El gestor de acciones es el subsistema encargado de la creación y borrado de las entidades modeladas por la aplicación. De esta forma, el gestor de acciones se encarga de la interacción con la base de datos con el fin de realizar modificaciones en la misma que afecten a productos, modelos de precio y contratos. Además, este gestor es el encargado de interpretar y modificar la información contenida en forma de XML para cada una de estas entidades.

El gestor de acciones, por tanto, está constituido por cuatro gestores específicos que se encargarán de manejar productos, modelos de precio y contratos respectivamente. A continuación, se describe brevemente la funcionalidad con la que cuenta cada gestor, así como las consideraciones especiales que dicha funcionalidad tiene.

6.4.1. Gestor de productos

Este gestor proporciona a las capas superiores la funcionalidad de definir y eliminar productos asegurando siempre la coherencia con el resto de elementos. El borrado de un producto supone automáticamente el borrado de su modelo de precio. Además, supone también la cancelación de todos los contratos (sin penalización económica para el cliente) que pesen sobre él realizándose una facturación automática de los mismos.

Esta funcionalidad proporciona a los proveedores de servicios la posibilidad de gestionar productos, únicamente indicando las características de los mismos. Esto se realiza a través de una interfaz de usuario únicamente accesible a proveedores que obtiene las características a través de formularios dinámicos.

6.4.2. Gestor de modelos de precio

El gestor de precios proporciona la funcionalidad de definir modelos de precio. El borrado de un modelo de precio es siempre consecuencia del borrado del producto asociado al mismo. La funcionalidad de definición de modelo de precio siempre se invoca desde la definición de un nuevo producto.

La definición de un modelo de precio conlleva una comprobación de integridad previa con el producto cuyo precio modela. De esta manera, no se introducen en el sistema modelos de precio incoherentes con sus productos.

Esta funcionalidad proporciona a los proveedores de servicios la posibilidad de gestionar el modelo de precio asociado a un producto indicando únicamente las características del mismo. La definición de un modelo de precio se realiza a través de una interfaz de usuario que obtiene dichos parámetros de un formulario dinámico que rellenará el usuario.

6.4.3. Gestor de contratos

Este gestor proporciona la funcionalidad de contratación y cancelación de contrato a los clientes del sistema. Para la contratación es necesario únicamente proporcionar al gestor las características del contrato, elegidas de entre las opciones del modelo de precio. Se comprueba que dichas características corresponden a las opciones ofrecidas para evitar inconsistencias. La cancelación genera información de uso para este contrato que permite facturar la penalización correspondiente.

Esta funcionalidad se invoca desde una interfaz de usuario accesible para los clientes que permite seleccionar las características deseadas para el contrato a través de selectores de opción múltiple, así como cancelar un contrato seleccionando de un listado.

6.4.4. Gestor de facturación

El gestor de facturación es el encargado de gestionar el proceso de facturación. Este proceso es el encargado de transformar los dato de uso en una factura. Se ejecuta automáticamente con el vencimiento del periodo de facturación para cada contrato.

A continuación, se crea una alarma que invoca el proceso de facturación tras el vencimiento de cada periodo de cobro. Dicha alarma invoca la facturación del contrato cuyo periodo ha vencido según los datos de uso almacenados para el mismo. Estos datos de uso incluyen interacciones del cliente con el servicio, información de incumplimiento del acuerdo de nivel de servicio e información de una cancelación del contrato si la hubiera.

Toda esta información, combinada con la información del contrato y del modelo de precio asociado al producto, es utilizada por el proceso de facturación. Este proceso consta de tres actividades principales: particionamiento de los datos de uso, cálculo de precios y generación de factura, que se detallarán a continuación.

Además, el proceso de facturación puede ser ejecutado de forma manual por los usuarios para conocer la facturación parcial intermedia de dicho contrato. La generación manual de una factura parcial no afecta a la facturación automática al final del periodo, manteniéndose disponibles todos los datos de uso para esta. Tras la realización de una facturación automática los datos de uso se deshabilitan de forma que no formen parte de otra facturación automática posterior.

Particionamiento de los datos de uso

El procesamiento de los datos de uso se rige por condiciones sobre los mismos. En el caso de datos temporales, estas condiciones pueden ser sobre la duración (actual o total) o sobre el momento de uso. Esto hace que un mismo dato de uso puede verse afectado por una condición

sólo parcialmente, por lo que será necesario tarificar dicho dato de uso por tramos según distintas condiciones.

El objetivo del particionamiento de los datos de uso es transformar los datos de uso en otros equivalentes mediante la división de los mismos, de forma que ningún dato esté parcialmente contenido en una condición, ya sea temporal o de uso acumulado (actual o total). El proceso de particionamiento examina el modelo de precio y el contrato para obtener las condiciones de tarificación que afectan a los datos de uso de cada componente del producto contratado.

El particionamiento de los datos de uso conlleva un problema adicional: al particionar un dato de uso en varios, el uso acumulado actual varía si se consideran los nuevos datos de uso por separado. Es por esto que se requiere identificar el dato de uso original de los datos de uso particionados con el fin de poder comprobar si su origen es el mismo y tarificarlos juntos si fuera necesario.

Además de esto, es también necesario identificar de forma unívoca cada nuevo dato de uso particionado. Esto permite identificar las tarificaciones del mismo dato de uso por efecto de distintos elementos de precio con el objetivo de almacenar finalmente aquella que tenga menor precio.

Cálculo de precios para la facturación

El proceso de cálculo de precios genera un conjunto de líneas de facturación a partir de los datos de uso procesados, el contrato y el modelo de precio asociados a un producto. Estas líneas de facturación representan distintos elementos a facturar junto al precio calculado para los mismos. Estos elementos pueden consistir en una suscripción, un uso determinado realizado por el usuario (puntual o a lo largo de un intervalo temporal), la cancelación del contrato, el incumplimiento de un acuerdo de nivel de servicio o un límite superior o inferior impuesto a la factura.

Para la obtención de estas líneas de facturación, cada dato de uso particionado se procesa junto a cada elemento de precio aplicable al mismo, obteniéndose así varias líneas de facturación asociadas al mismo dato de uso. A continuación, se procesan estas líneas de facturación para almacenar definitivamente únicamente la de menor precio.

Generación de factura

El proceso de generación de factura es el que, a partir de las líneas de facturación obtenidas previamente, genera un XML de salida y una factura en formato PDF. El formato específico del XML de salida se especifica posteriormente. Este XML contiene todas las líneas de facturación obtenidas para un contrato además de otra información adicional.

Mediante el uso de la herramienta JasperReports[20], el contenido del XML de factura se transforma en un fichero en formato PDF que detalla cada línea de factura y el total de la misma. Este fichero PDF podrá ser visualizado por el cliente desde la interfaz del sistema.

6.5. Interfaz de usuario

El sistema cuenta con una interfaz web de usuario para la realización de las acciones definidas en los casos de uso e implementadas en los gestores. Cada elemento de la interfaz gestiona la creación, visualización o borrado de alguno de los elementos del sistema (productos, modelos de precio y contratos) ofreciendo a los usuarios una forma visual e intuitiva de interactuar con el sistema. Los distintos elementos y las maquetas correspondientes a ellos pueden consultarse en el Anexo B.

6.6. Definición de los datos de entrada

Los datos de entrada de la aplicación representan los datos de uso de los distintos productos contratados y son almacenados por aplicaciones externas a este tarificador para su posterior facturación. Estos datos están representados en una simplificación del formato SDR (Service Detail Record) de monitorización de comunicaciones. Los datos de uso almacenados en este formato pueden representar información relativa a eventos, cantidades e intervalos de tiempo. El formato SDR está definido en lenguaje JSON[21] y consta de tres tipos de ficheros con tres estructuras distintas: de eventos, de cantidades y temporales.

A pesar de tratarse de información externa al tarificador, se proporcionan a los proveedores de servicios métodos que abstraen el almacenamiento de esta información y su escritura en formato SDR.

Se puede consultar la estructura de los ficheros SDR en el anexo E

Este formato se adecúa a los datos de entrada de esta aplicación ya que permite un almacenamiento ágil de la información de uso en la base de datos en tiempo real al estar basado en JSON.

6.7. Definición de los datos de salida

Los datos de salida de la aplicación, que representan cada una de las facturas obtenidas, se almacenan como ficheros XML con formato CDR[22]. Este formato ha sido definido por la 3rd Generation Partnership Project (3GPP)[23], que tiene como objetivo crear especificaciones para las comunicaciones de tercera generación (3G). Se trata de un formato de almacenamiento de información relativa a cobros de eventos de telecomunicaciones. Los CDRs se usan en compañías

de telecomunicación para transferir información de facturación. Se utilizará un subconjunto de la funcionalidad definida en este formato que permitirá transmitir la información detallada de cada cobro manteniendo la simplicidad. En el anexo F puede verse la definición como esquema XSD[16] del subconjunto del formato CDR que se va a utilizar junto con un ejemplo de un fichero en este formato.

La información representada en este XML se procesa para la creación de una factura en PDF mediante el uso de la herramienta JasperReports[20].

7

Implementación

7.1. Introducción

A continuación se detalla cómo se han implementado los distintos componentes descritos en el capítulo anterior. Se indican las herramientas que se han utilizado para su implementación y se explican los algoritmos específicos que han permitido implementar la funcionalidad definida.

7.2. Base de datos

La base de datos previamente definida como diagrama entidad-relación se ha implementado en lenguaje MySQL[11]. Para la transformación de la base de datos del modelo Entidad-Relación al modelo relacional se ha seguido el algoritmo definido por Elmasri [24].

7.3. Estructura de la aplicación

El código de la aplicación conecta con el modelo de datos implementado como base de datos MySQL[11] e implementa también la funcionalidad de la aplicación sobre el mismo. Además, presenta esta funcionalidad en forma de interfaz gráfica para que los usuarios puedan hacer uso de la misma.

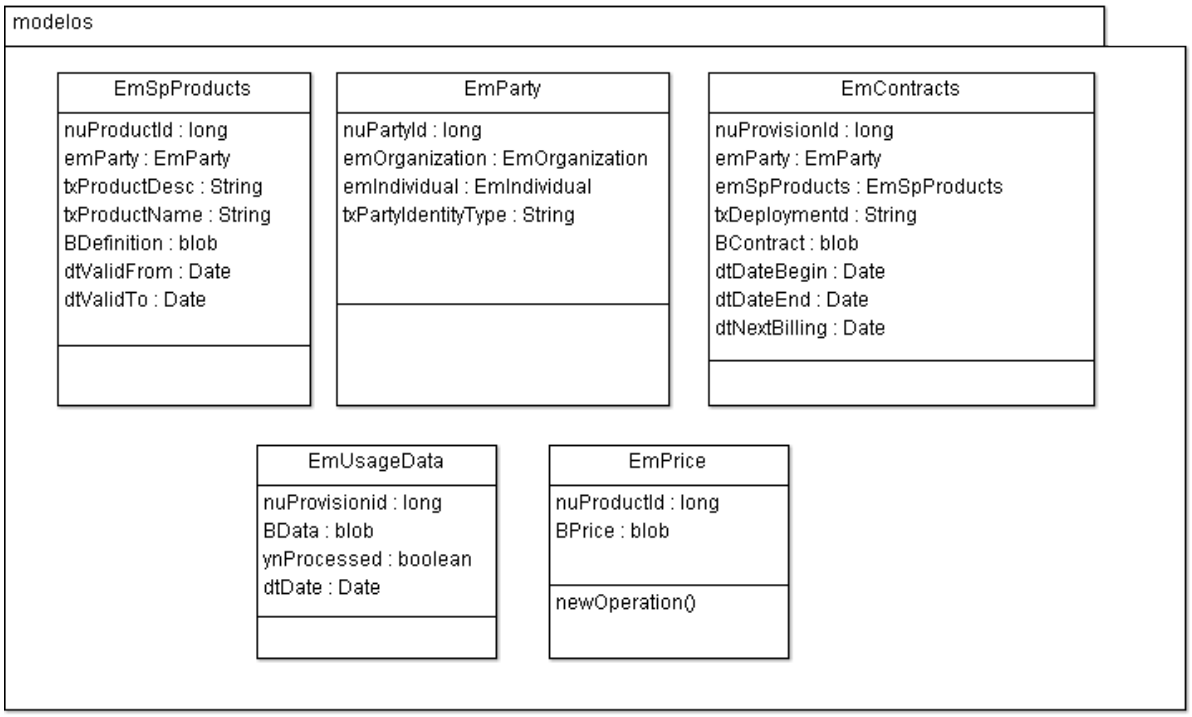

Figura 7.1: Fragmento del diagrama de clases correspondiente al código del paquete modelos.

Se ha decidido implementar esta aplicación en el lenguaje Java que permite la representación de las distintas entidades manejadas por la aplicación como clases. Este lenguaje, altamente extendido, cuenta por tanto con múltiples plataformas de soporte de distintas tareas como conexión y manejo de bases de datos, interpretación de ficheros XML[15] y JSON[21], diseño de interfaces según el Modelo Vista Controlador y generación de ficheros PDF.

La funcionalidad implementada en el código de esta aplicación incluye el acceso a la base de datos y a los datos contenidos en la misma, el tratamiento de dichos datos, la implementación del proceso de facturación, la interpretación y gestión de los datos de entrada y de salida y el control de la interfaz de usuario.

Se ha realizado un diagrama de clases UML que representa la estructura de las clases de la aplicación. Cada paquete se muestra y explica a continuación.

7.3.1. Conexión con la base de datos

En las figuras 7.1 y 7.2 se muestran los paquetes *modelos* y *dao*. Estos paquetes son los relacionados con la conexión con la base de datos, que se ha realizado mediante el uso de la herramienta Hibernate[12] de mapeo objeto-relacional para Java.

El paquete *modelos* contiene las representaciones de las entidades definidas en la base de datos en lenguaje java. El paquete *dao* a su vez contiene las clases manejadoras de dichas entidades

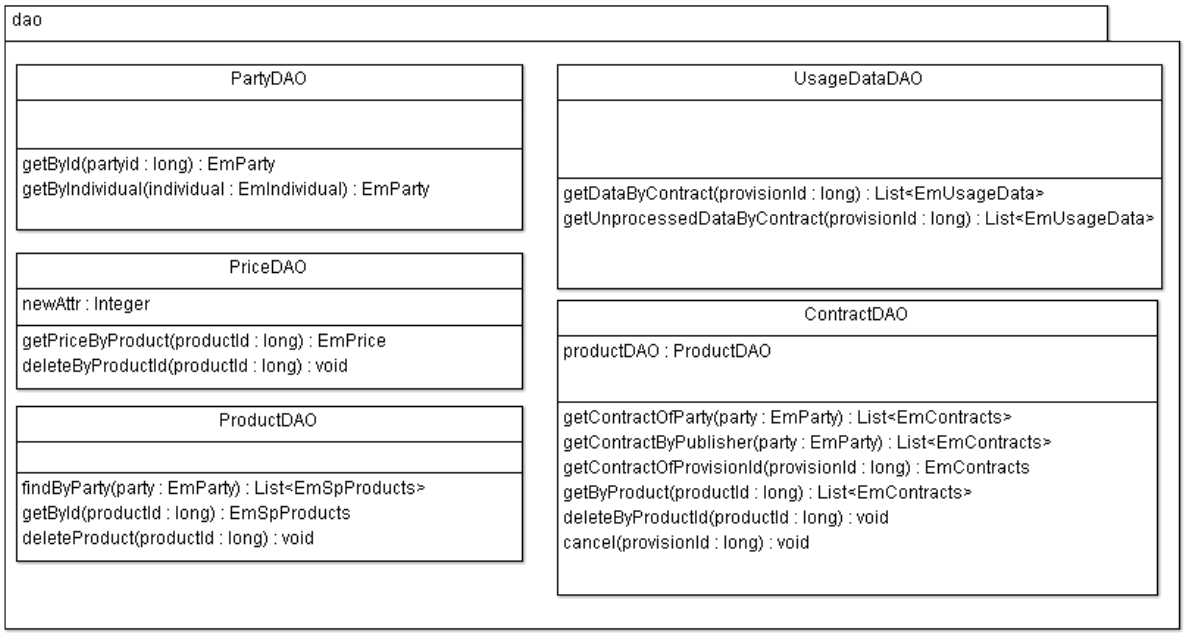

Figura 7.2: Fragmento del diagrama de clases correspondiente al código del paquete dao.

que permiten la búsqueda, inserción y borrado de los mismos. En la figura 7.2 pueden además verse los métodos adicionales que se han implementado para la búsqueda, inserción o borrado por distintos criterios para su uso en la aplicación.

7.3.2. Gestores del modelo de datos

En la figura 7.3 se muestran los gestores de la aplicación. Se encargan de gestionar las distintas entidades representadas en la base de datos así como las distintas acciones o casos de uso de la aplicación. Las clases ContractManager, ProductManager y PriceManager se encargan de la creación, edición, borrado y cancelación de los contratos, los productos y los modelos de precio. Además, se incluye el método de comprobación de coherencia entre productos y modelos de precio *checkPriceProduct*.

PartyManager es el gestor encargado de las acciones relativas a los usuarios. El tarificador hace uso de dicha clase para comprobar los tipos de moneda aceptados por un usuario.

UsageDataManager es la clase encargada de la gestión de los datos de uso. Esta gestión consiste en el procesamiento de los mismos mediante el algoritmo de particionamiento de los datos de uso temporales que se explica en la sección 7.4. Esta clase define distintos métodos auxiliares que son usados por el método principal *timeDataProcess*. Además, esta clase ofrece métodos para la inserción de los datos de uso por parte de los proveedores de servicios. Estos métodos permiten a los proveedores de servicios abstraerse del lenguaje en que estos datos de uso están representados.

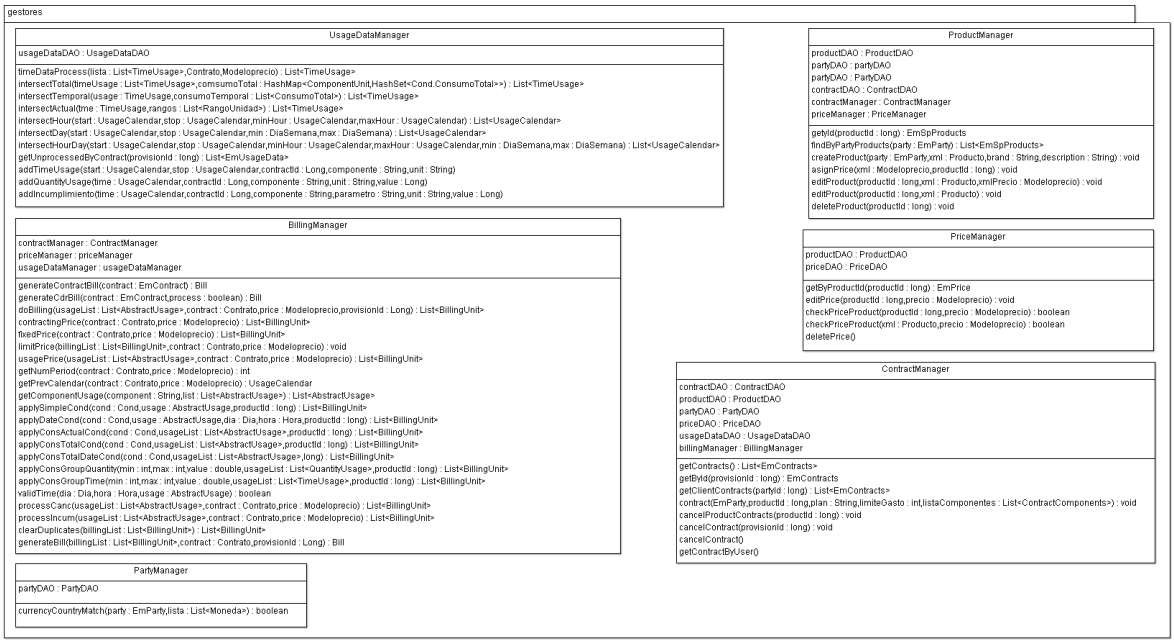

Figura 7.3: Fragmento del diagrama de clases correspondiente al código del paquete gestores.

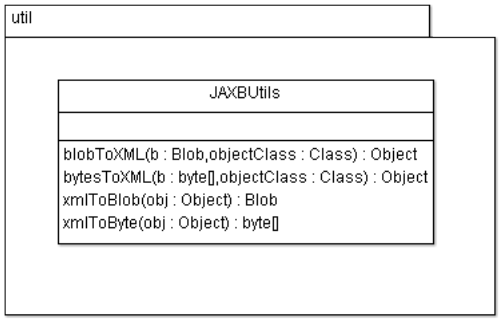

Figura 7.4: Fragmento del diagrama de clases correspondiente al código del paquete util.

BillingManager es la clase encargada de realizar la facturación de los datos de uso con la información del modelo de precio y el contrato asociados. Para ello, se definen métodos auxiliares que serán usados por los métodos principales de la clase *generateContractBill* y *generateCdrBill*. El método *generateContractBill* es llamado tras la contratación para generar una factura por el precio inicial de la misma. El método *generateCdrBill* es llamado para realizar la facturación, tanto manual como automática. Recibe como parámetro un valor booleano que le permite determinar de qué tipo de facturación se trata y, consecuentemente, marcar los datos de uso utilizados como procesados o no.

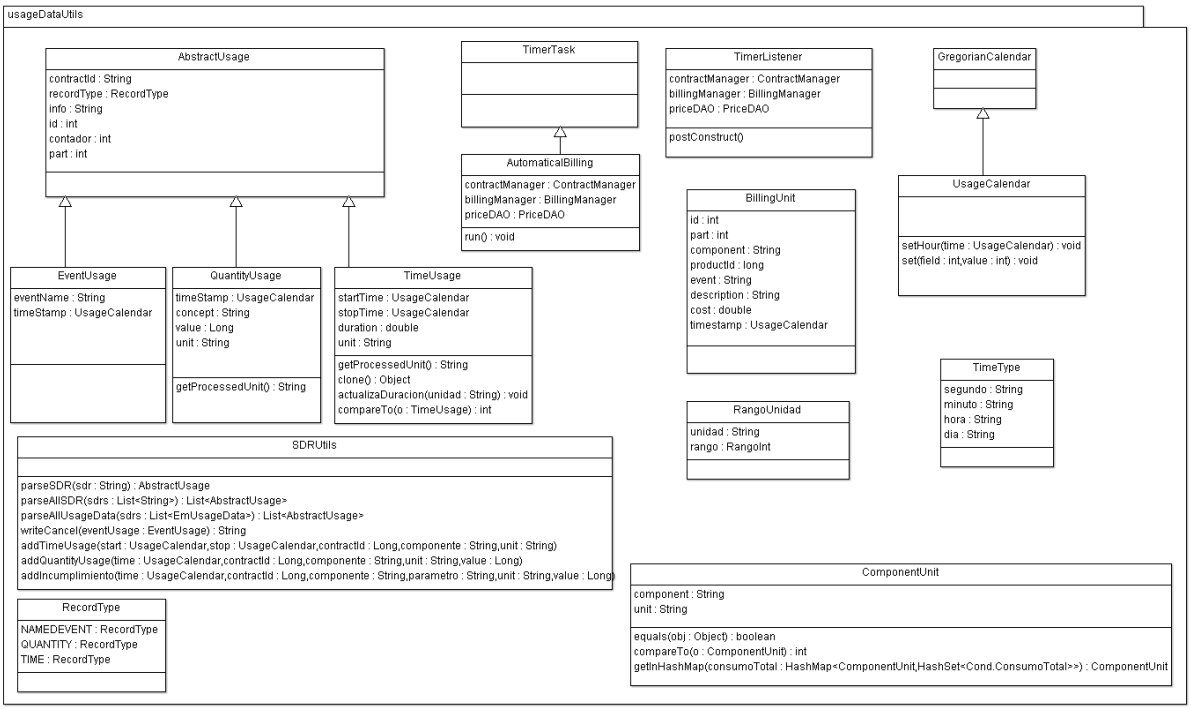

Figura 7.5: Fragmento del diagrama de clases correspondiente al código del paquete usageDataUtils.

7.3.3. Tratamiento auxiliar de datos y cronómetro de facturación

En las figuras 7.4, 7.5 y 7.6 se muestran los paquetes *util*, *usageDataUtils* y *xmlUtils*. Estos paquetes contienen clases auxiliares de la aplicación para, por ejemplo, el tratamiento de datos.

util

El paquete *util* cuenta con la clase *JAXBUtils*, de transformación de datos binarios a su representación XML en Java y viceversa. Los ficheros XML se almacenan en la base de datos como archivos binarios de tipo *Byte* o *Blob*. Para la transformación de estos datos en su clase XML manejadora y viceversa, se han definido 4 métodos (2 correspondientes a la transformación para *Byte* y 2 correspondientes a la transformación para *Blob*).

usageDataUtils

Este paquete cuenta con clases representantes de pares de elementos o enumerados para su uso durante el proceso de facturación. Además, cuenta con las siguientes clases esenciales para el particionamiento de los datos de uso y el proceso de facturación.

UsageCalendar Esta clase, que extiende a la clase *GregorianCalendar*, redefine métodos en la misma para la modificación de valores específicos del calendario.

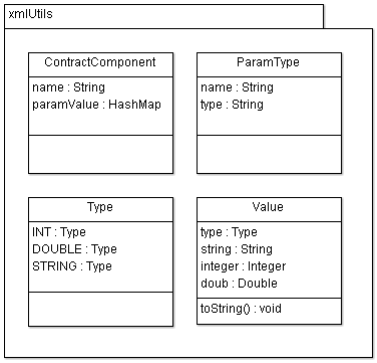

Figura 7.6: Fragmento del diagrama de clases correspondiente al código del paquete xmlutils.

TimerListener Esta clase, cuyo método *postConstruct* se ejecuta tras iniciar la aplicación, se encarga de inicializar el cronómetro de la aplicación. Este cronómetro con periodo de un día, ejecuta el método *run* de la clase *AutomaticalBilling* para la facturación automática. El cronómetro se inicializa en el día de la ejecución a las *00:00* horas, de forma que siempre se realice una ejecución del método *run* al iniciar la aplicación.

AutomaticalBilling Esta clase, que extiende la clase *TimerTask* realiza las comprobaciones para la facturación automática. El método *run* de la misma se ejecuta cada vez que termina un ciclo del cronómetro de la aplicación. Al ejecutarse, el método comprueba, para cada contrato existente, si su fecha de próxima facturación coincide con la fecha actual. Si así fuera, se realizaría una facturación automática del mismo.

BillingUnit Clase de representación de cada una de las líneas de facturación durante el proceso de facturación. Tras finalizar éste, estas líneas de facturación son transformadas en un fichero CDR de factura.

SDRUtils Esta clase tiene como función la de transformar todos los datos de uso en formato SDR en objetos Java con formato específico. Además, también ofrece métodos para la escritura de datos SDR, incluyendo la información de cancelación.

AbstractUsage Esta clase y las tres clases que heredan de ella (*QuantityUsage*, *EventUsage* y *TimeUsage*), representan los tres tipos de datos de uso existentes en la aplicación y definidos en formato SDR. Tras el procesamiento de los datos de uso para su uso en Java, cada uno de ellos se almacena como un objeto de alguna de las clases herederas.

xmlUtils

Este paquete cuenta con clases representantes de pares o ternas de elementos, para su posterior asignación a un fichero XML. Además, cuenta con la clase *Value* que permite representar de forma abstracta los tres tipos de datos que puede tener un parámetro dentro de un producto (cadena de texto, entero o real).

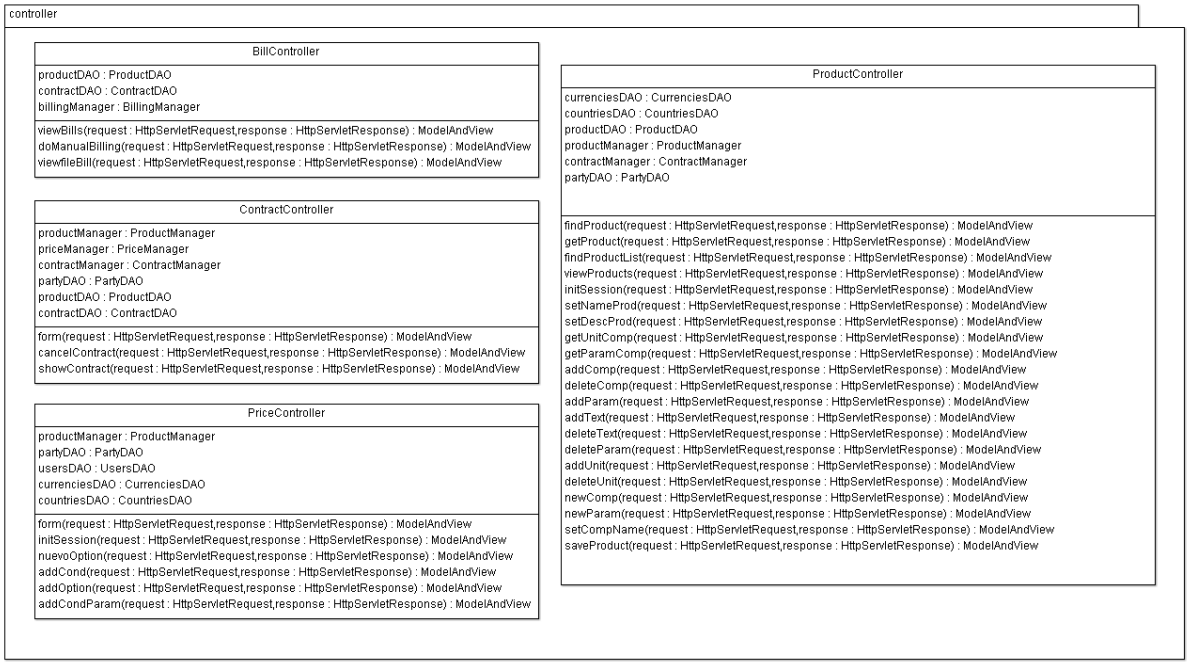

Figura 7.7: Fragmento del diagrama UML correspondiente a la interfaz de usuario.

7.3.4. Controladores de la interfaz de usuario

En la figura 7.7 se muestra el paquete contenedor de las clases controladoras de la aplicación. Se trata de las clases del servidor de la aplicación para la interfaz de usuario del mismo, que sigue la estructura vista-controlador del Framework Spring. Cada una de estas clases controla un proceso o entidad de la aplicación, como productos o facturación. Cada uno de sus métodos corresponde a una acción a realizar en la interfaz, y su ejecución se solicita a través de una petición HTTP POST o mediante Ajax.

7.3.5. Interpretación de los datos de entrada

Aunque aquí no se muestran mediante un diagrama de clases dada su estructura, el código cuenta con clases JAXB generadas automáticamente. Estas clases representan los ficheros XML específicos definidos para los datos de entrada de la aplicación y se generan mediante la herramienta JAXB[17] a partir de sus esquemas XSD de definición. Estas clases mantienen una estructura de árbol con subclases incluidas en ellas para representar la estructura de los ficheros XML.

7.4. Algoritmo de particionamiento

El algoritmo de particionamiento es el encargado de transformar los datos de uso en otros equivalentes mediante la división de los mismos para que ningún dato quede parcialmente contenido en una condición temporal o de uso acumulado.

El particionamiento de los datos de uso conlleva un problema adicional: al particionar un dato de uso en varios, el uso acumulado actual variará si se consideran los nuevos datos de uso por separado. Es por esto que se requiere identificar el dato de uso original de los datos de uso particionados con el fin de poder comprobar si su origen es el mismo y tarificarlos juntos si fuera necesario. Además de esto, es también necesario distinguir dos datos de uso, ya sean originales o particionados, si no se refieren al mismo uso en el mismo instante. Esto permite identificar las tarificaciones del mismo dato de uso por efecto de distintas condiciones para almacenar únicamente una de ellas.

Para realizar lo anterior, se asigna un identificador único incremental a cada dato de uso durante su lectura y un identificador particular único para cada parte del mismo. Este identificador único se asigna como 1 al dato original se incrementa en los datos obtenidos en cada expansión asignándose como $p_{n,i} = T_{n-1}p_{n-1} + i$ donde p_{n-1} es el valor asignado al dato particionado, $p_{n,i}$ es el valor de la iésima parte del mismo y T_n es el total de datos obtenidos en la expansión n-ésima.

El algoritmo de particionamiento examina el modelo de precio y el contrato para obtener las condiciones de tarificación que afectan a los datos de uso de cada componente del producto contratado. Estas condiciones se obtienen tanto de entre las condiciones generales del plan elegido como de las condiciones particulares según los parámetros seleccionados durante la contratación.

Las condiciones se aplican únicamente a aquellos datos de uso del componente al que dichas condiciones pertenecen y siempre que la unidad de uso coincida.

El particionamiento se realiza en tres fases consecutivas que transforman los datos de uso obtenidos de la fase anterior. Cada particionamiento genera una lista de instantes en el tiempo que son los extremos de los intervalos que se van a definir. Al finalizar parcial o totalmente un particionamiento se transforman dichos instantes en intervalos mediante una ordenación previa. De esta forma, si los instantes obtenidos ordenados cronológicamente de forma creciente son $\{t_1, \cdots, t_n\}$, los intervalos en los que se recompondrán serán $\{(t_1, t_2), \cdots, (t_{n-1}, t_n)\}$. A continuación se definen y explican estas fases y el particionamiento realizado en las mismas.

7.4.1. Particionamiento por uso actual acumulado

El particionamiento por uso actual acumulado particiona cada dato de uso según los tramos definidos por las condiciones de uso acumulado. De este modo, aquellos tramos del uso donde la

duración del mismo se encuentre dentro de la franja definida por alguna condición de este tipo se definirán como un nuevo uso, respetando la identificación definida para los usos particionados.

Este particionamiento se realiza el primero ya que, al no haberse realizado ningún particionamiento previo, son más sencillos los cálculos sobre el uso acumulado.

Para cada uso (t_{inicio}, t_{fin}), si el conjunto de condiciones que lo afectan define los intervalos de uso acumulado $\{(a_{1min}, a_{1max}), \cdots, (a_{nmin}, a_{nmax})\}$ en segundos, minutos, horas o días, el conjunto final de instantes con los que particionar será

$$\{t_{inicio}, t_{fin}\} \cup \{t : t = t_{inicio} + a_{imin}, t < t_{fin}\} \cup \{t : t = t_{inicio} + a_{imax}, t < t_{fin}\}.$$

Dicho conjunto se recompondrá en intervalos, de la forma que se ha explicado previamente, antes de realizar el siguiente particionamiento.

7.4.2. Particionamiento por condiciones temporales

El particionamiento por condiciones temporales particiona según las condiciones de momento de uso o uso total acumulado en un intervalo temporal. Como se ha visto, estos dos tipos de condiciones se definen de la misma forma y será el particionamiento por uso total acumulado posterior el que tenga en cuenta las diferencias entre ambos.

El particionamiento se realiza de forma recursiva, aplicando en cada ejecución una condición. Tras cada ejecución, una vez obtenidos los nuevos intervalos de uso, se realizará una nueva llamada recursiva eliminando la condición usada. La condición de parada de esta recursión es un conjunto vacío de condiciones a aplicar.

Este particionamiento tiene tres formas distintas de realizarse según si la condición es horaria, diaria u horaria y diaria al mismo tiempo. La lista de instantes por los que particionar se obtiene como se explica a continuación para cada uno de ellos.

Particionamiento horario

El particionamiento horario divide los datos de uso separándolos por las horas de inicio y fin del intervalo en cualquier día. Por tanto, si el uso durase varios días se realizarían particionamientos por las horas de inicio y fin en cada uno de ellos.

Si el uso a particionar es de la forma (t_{inicio}, t_{fin}) y el intervalo horario de la condición es (t_{min}, t_{max}), el conjunto de instantes por los que particionar obtenido en este proceso será

$$\{t_{inicio}, t_{fin}\} \cup \{t : t.hora = t_{min}, t_{inicio} < t < t_{fin}\} \cup \{t : t.hora = t_{max}, t_{inicio} < t < t_{fin}\}.$$

Estos instantes se recomponen en intervalos dentro del algoritmo de particionamiento temporal general. Nótese que no surge ningún problema si $t_{min} > t_{max}$.

Particionamiento diario

El particionamiento diario divide los datos de uso separándolos según el intervalo de una condición expresado en días de la semana. El particionamiento se realiza en cada cambio de día de la semana que suponga un cambio de la condición respecto al día anterior. Esto podría producir varios instantes de cambio para un mismo día de la semana en un uso que durase más de una semana.

Si el uso a particionar es de la forma (t_{inicio}, t_{fin}) y el intervalo diario de la condición es (t_{diaMin}, t_{diaMax}), el conjunto de instantes por los que particionar obtenido en este proceso será

$\{t_{inicio}, t_{fin}\} \cup \{t : t.dia = t_{diaMin}, t.hora = 00:00, t_{inicio} < t < t_{fin}\} \cup \{t : t.dia = t_{diaMax} + 1, t.hora = 00:00, t_{inicio} < t < t_{fin}\}$.

Estos instantes se recomponen en intervalos dentro del algoritmo de particionamiento temporal general. Nótese que no surge ningún problema si $t_{diaMin} > t_{diaMax}$ tomando el domingo como día inicial de la semana.

Particionamiento horario y diario

Las condiciones temporales con día y hora son válidas cada día de la semana de entre los definidos en el intervalo diario durante las horas definidas en el intervalo horario. Esto significa que el particionamiento se realizará por cada uno de los instantes de entrada en vigor o cese de efecto de la condición: cada día de la semana de entre los definidos a las horas mínima y máxima dentro del periodo del uso.

Si el uso a particionar es de la forma (t_{inicio}, t_{fin}),el intervalo diario de la condición es (t_{diaMin}, t_{diaMax}) y el intervalo horario es $(t_{horaMin}, t_{horaMax})$, el conjunto de instantes por los que particionar obtenido en este proceso será

$\{t_{inicio}, t_{fin}\} \cup \{t : t.hora = t_{horaMin}, t_{diaMin} < t.dia < t_{diaMax}, t_{inicio} < t < t_{fin}\} \cup \{t : t.hora = t_{horaMax}, t_{diaMin} < t.dia < t_{diaMax}, t_{inicio} < t < t_{fin}\} \cup \{t : t.dia = t_{diaMin}, t.hora = 00:00, t_{inicio} < t < t_{fin}\} \cup \{t : t.dia = t_{diaMax} + 1, t.hora = 00:00, t_{inicio} < t < t_{fin}\}$.

Estos instantes se recomponen en intervalos dentro del algoritmo de particionamiento temporal general.

Para la comparación de días de la semana se realiza una comparación modular donde si tomando el día inicial de la semana como domingo, si $t_{diaMin} > t_{diaMax}$, $t_{diaMin} < t.dia < t_{diaMax}$ es equivalente a $t_{diaMin} < t.dia < t_{diaMax} + 7$ o $t_{diaMin} - 7 < t.dia < t_{diaMax}$.

7.4.3. Particionamiento por uso total acumulado

El particionamiento por uso total acumulado particiona según el total acumulado de todos los datos de uso sobre un componente con una determinada unidad, o sobre todos aquellos que además de lo anterior satisfagan una condición temporal.

Este particionamiento se realiza el último para trabajar sobre los resultados del particionamiento temporal. Dado que ya se ha realizado dicho particionamiento temporal, ningún dato de uso satisfará una condición temporal parcialmente.

Para cada condición se obtiene el conjunto de datos de uso a los que afecta que puede ser el total si carece de condición temporal o un subconjunto de los datos de uso obtenidos de los particionamientos anteriores que se encuentren dentro del periodo indicado.

Una vez obtenidos, estos datos de uso se ordenan cronológicamente y, si los datos de uso son de la forma $\{(t_{1inicio}, t_{1fin}), \cdots, (t_{ninicio}, t_{nfin})\}$ y el intervalo de la condición es (t_{min}, t_{max}) se obtiene como conjunto de instantes de particionamiento

$\{t_{1inicio}, t_{1fin}, \cdots, t_{ninicio}, t_{nfin}\} \cup \{t : \exists k, t - t_{kinicio} + \sum_{i=1}^{k-1} t_{ifin} - t_{iinicio} = t_{min}\}$

$\cup \{t : \exists k, t - t_{kinicio} + \sum_{i=1}^{k-1} t_{ifin} - t_{iinicio} = t_{max}\}$.

Estos instantes se recomponen en intervalos dentro del algoritmo de particionamiento temporal general.

7.5. Algoritmo de facturación inicial

Cada modelo de precio tiene definido un precio inicial para el producto asociado al mismo. Además, puede tener un precio inicial asociado a la elección de una determinada opción para un determinado parámetro. El precio inicial total se factura al cliente inmediatamente tras contratar dicho producto. Este precio es la suma de los costes iniciales del plan elegido y de los parámetros seleccionados según su valor.

Para el cálculo de dicha factura, el algoritmo toma el modelo de precio y el contrato realizado. A partir de este último, obtiene información acerca del plan y los parámetros seleccionados. A continuación, obtiene del modelo de precio el valor de cobro inicial asociado a éstos y añade una línea de facturación por cada uno de ellos.

7.6. Algoritmo de facturación periódica

Cada producto tiene definido en su modelo de precio un periodo de facturación que puede ser de uno o varios días, semanas o meses. Cada vez que transcurre el tiempo indicado en dicho periodo, se produce un vencimiento del periodo de facturación que produce automáticamente

una factura a partir del modelo de precio del producto, el contrato realizado y los datos de uso del producto en ese periodo. Esta facturación también puede realizarse de forma manual en cualquier momento, con la única diferencia de que la facturación manual muestra únicamente una factura parcial del periodo según los datos de uso existentes hasta el momento de la misma. Además, la facturación manual no invalida los datos para una facturación (automática o manual) posterior, mientras que las facturaciones automáticas sí lo hacen, evitando que los datos de uso se facturen en más de un periodo.

El objetivo del algoritmo de facturación es la obtención de un conjunto de líneas de facturación a partir de los datos de uso, el contrato y el modelo de precio asociados a un producto. Estas líneas de facturación representan distintos elementos a facturar junto al precio calculado para los mismos. Estos elementos pueden consistir en una suscripción, un uso determinado realizado por el usuario (puntual o a lo largo de un intervalo temporal), la cancelación del contrato, el incumplimiento de un acuerdo de nivel de servicio o un límite superior o inferior impuesto a la factura.

El algoritmo de facturación consiste en el procesamiento secuencial de los datos de uso, el contrato y el modelo de precio a través de distintos procesos que generan líneas de facturación según distintos criterios. Cada dato de uso puede procesarse varias veces si existen varios criterios de facturación que sean aplicables al mismo. De todas las líneas de facturación generadas para un mismo dato de uso, finalmente sólo se almacenará la de menor precio.

7.6.1. Facturación de precio fijo por periodo

Cada producto tiene asignado un precio para cada periodo de cobro. Este precio, que puede variar en función del periodo, es independiente del uso realizado. Además, cada una de las opciones disponibles para cada parámetro durante la contratación, puede tener un precio por periodo suplementario definido. Esto quiere decir que si se selecciona dicha opción, además de la cantidad general, se facturará también ésta. Para cada uno de los precios por periodo aplicables al contrato, el algoritmo generará una línea de facturación.

Para obtener el valor de cada precio periódico aplicable en el periodo actual, se obtiene el periodo actual como $\frac{hoy - inicio\_del\_contrato}{longitud\_de\_cada\_periodo}$ donde todos los datos están medidos según el tipo de periodo (día, semana, mes o año). A continuación, habiendo obtenido el número de periodo actual, se consulta el modelo de precio para obtener, para cada precio periódico, el precio fijo asignado al mismo.

En el anexo D puede verse un ejemplo de facturación periódica de un producto según su modelo de precio.

7.6.2. Procesamiento de los datos de uso

Este procesamiento tiene como objetivo facturar el uso de un producto durante el periodo actual según las reglas o condiciones definidas en el modelo de precio que afecten al contrato. Estas condiciones son aquellas contenidas en el plan seleccionado por el usuario, además de aquellas asociadas a los valores de parámetros que se hayan seleccionado durante la contratación.

Para realizar esta facturación, se toman los datos de uso del usuario, que deben haber sido almacenados por la aplicación contratada. Estos datos de uso están siempre asociados a un componente. Por tanto, el proceso de facturación de datos, tras obtener las condiciones que afectan a dicho componente según el contrato, obtiene los datos de uso asociados al mismo. Una vez obtenidos dichos datos de uso, cada condición es procesada junto con todos los datos de uso del componente, según su tipo.

Procesamiento de tarifas incondicionales

Las tarifas incondicionales siempre se aplican a todos los datos de uso a los que afectan. Por cada dato de uso, se crea una unidad de facturación cuyo precio sea:
$tarifa \cdot cantidad\ del\ dato\ de\ uso$
Si el dato de uso es temporal, esta cantidad se refiere a la duración del mismo.

Procesamiento de tarifas aplicables a un intervalo temporal

El procesamiento de tarifas temporales es idéntico al de tarifas incondicionales excepto por una comprobación inicial que elimina del procesamiento aquellos datos de uso que no se encuentran en el intervalo temporal especificado por la tarifa. Dado que los datos de uso ya han sido particionados, todos ellos se encuentran íntegramente dentro de los límites de la tarifa o completamente fuera de los mismos. Por tanto, únicamente se comprueba que el día y la hora de comienzo del uso (y de finalización en caso de tratarse de un uso temporal) se encuentran en el intervalo definido por la tarifa. Para que esto se satisfaga, en caso de existir una condición sobre el día de uso, debe cumplirse lo siguiente: $diamin \leq diamax \wedge diamin \leq diainicio \wedge diafin \leq diamax$ ó $diamin > diamax \wedge (diamin \leq diainicio \vee diafin \leq diamax)$

Procesamiento de tarifas sobre el consumo actual

El procesamiento de tarifas sobre el consumo actual depende del consumo durante un mismo uso. Dado que los datos de uso temporales han sido particionados, es necesario recuperar todas las partes de un mismo dato de uso original. Estas partes se ordenan cronológicamente desde la más antigua a la más reciente. A continuación, para cada una de ellas se comprueba si se

satisface la condición mediante el uso de un contador. Dado que los datos de uso ya han sido particionados, todos ellos se encuentran íntegramente dentro o fuera de los límites de la tarifa.

A continuación se muestra un breve pseudocódigo simplificado de esta comprobación.

```
acumulado ← 0
for uso en lista do
  acumulado ← acumulado + uso.total
  if min ≤ acumulado ≤ max then
    facturar uso
  end if
end for
```

Procesamiento de tarifas sobre el consumo total

El procesamiento de tarifas sobre el consumo total es idéntico al de las tarifas sobre consumo actual excepto por el hecho de que se tienen en cuenta todos los datos de uso sobre un componente a la hora de realizar la comprobación. Estos datos de uso se ordenan cronológicamente de más antiguo a más reciente, y sobre ellos se ejecuta el algoritmo anterior. Dado que los datos de uso ya se han particionado, todos ellos se encuentran completamente dentro o fuera de los límites de la tarifa.

Procesamiento de tarifas sobre el consumo total en un intervalo temporal

El procesamiento de tarifas sobre el consumo total temporal toma los datos de uso que satisfacen las condiciones temporales siguientes:
$diamin \leq diamax \wedge diamin \leq diainicio \wedge diafin \leq diamax$ ó $diamin > diamax \wedge (diamin \leq diainicio \vee diafin \leq diamax)$
Sobre estos datos, se ejecuta el proceso de facturación de consumo total de la misma forma que en el caso anterior.

7.6.3. Facturación por cancelación

La cancelación prematura de un contrato por parte del usuario puede suponer el cobro de una penalización. Esta penalización puede componerse de una cantidad fija y/o una cantidad variable dependiente del tiempo restante de contrato en periodos. El proceso de facturación por cancelación localiza los datos de uso correspondientes a una cancelación, si hubiera alguno. En caso de detectar una cancelación, el proceso calcula el precio asociado a la misma, creando una unidad de facturación para el cobro fijo y otra para el cobro variable. Para el cálculo del cobro variable, se obtiene el número de periodos restantes como $duracion\ total - \frac{hoy - inicio\ del\ contrato}{longitud\ de\ cada\ periodo}$

medido en la unidad del periodo (días, semanas o meses), donde la duración total se refiere a la duración total del contrato en las mismas unidades. Una vez obtenido el número de periodos restantes, para cada uno de ellos se obtiene el precio asociado, facturándose la suma de los precios asociados a cada uno de los periodos, es decir,
$\sum_{i=n}^{restantes} precio_i$
En el anexo G puede verse un ejemplo de cancelación de un contrato.

7.6.4. Facturación por incumplimiento del acuerdo de nivel de servicio

Cada dato de uso indicando un incumplimiento se refiere a un parámetro de un componente para el cual no se ha ofrecido el nivel de calidad contratado durante un periodo de tiempo o durante la realización de varios usos puntuales. Dicho parámetro, en el plan elegido dentro del modelo de precio, puede tener asociados uno o varios precios a descontar al cliente en su factura. Para cada dato de uso referido a un incumplimiento, se localizan los precios asociados al mismo dentro del plan elegido en el modelo de precio. Cada uno de estos precios puede ser incondicional o depender de la magnitud del incumplimiento. Se generan unidades de facturación para cada uno de los precios incondicionales y para aquellos precios condicionales que satisfagan $min \leq magnitud \leq max$. Dichas unidades serán procesadas más adelante con el objetivo de almacenar únicamente la de coste menor.

7.6.5. Procesamiento de unidades de facturación duplicadas

Una vez obtenidas todas las unidades de facturación, se realiza una comprobación de las unidades duplicadas. Durante el algoritmo de particionamiento, se asignó a cada dato de uso particionado un identificador global según el dato de uso original y un identificador particular para cada nueva partición del mismo. Este par de identificadores, que se ha transferido a las unidades de facturación, se utiliza ahora para la detección de datos de uso facturados varias veces. Las unidades de facturación asociadas a cobros fijos o cancelación no están asociadas a un dato de uso y tienen asignado un par de identificadores especial que no será procesado.

Para ello, se crea una tabla hash cuyas claves son los pares de identificadores y cuyo contenido son conjuntos de unidades de facturación. Para cada clave asociada a datos de uso, se elige únicamente la unidad de facturación con precio menor, desechándose las demás.

7.6.6. Procesamiento de los límites de precio

Este procesamiento se realiza tras la eliminación de las unidades duplicadas para contar con un cálculo final del precio total de facturación. Se calcula dicho precio total como la suma de los costes de todas las unidades de facturación y se comprueba si aplica alguna de las condiciones límite.

Si existe un límite inferior (mínimo de precio), y $total < minimo$, se almacena una nueva unidad de facturación con este precio mínimo. Si existe un límite superior (máximo de precio), y $total > maximo$, se almacena una nueva unidad de facturación con este precio máximo. Si el usuario hubiese establecido un máximo de gasto, se comprobará que no se factura una cantidad superior a la establecida por él.

La unidades de facturación correspondientes a límites de precio serán procesadas más adelante durante el cálculo del total de facturación.

7.7. Algoritmos de comprobación de coherencia

Para el correcto funcionamiento de la aplicación, es imprescindible que los modelos de precio sean coherentes con los productos. Esto es, que todos los elementos como componentes o parámetros que éstos referencien, estén definidos en el producto. Es por esto que es necesario un algoritmo de comprobación de coherencia entre los distintos XML.

A continuación se describen las comprobaciones que realiza dicho algoritmo para garantizar la coherencia entre producto y modelo de precio.

7.7.1. Comprobación de coherencia sobre modelos de precio

El algoritmo de comprobación de coherencia sobre modelos de precio comprueba lo siguiente:

- De haber limitaciones máxima y mínima, $min < max$
- Cada componente está definido en el producto
- Cada parámetro está definido en el producto, dentro del componente indicado
- Cada parámetro toma valores según su tipo (entero, real o textual).
- Cada condición se define sobre unidades definidas para el componente.
- El incumplimiento se define sobre la unidad del parámetro.

7.8. Implementación de la interfaz

La interfaz de esta aplicación se ha implementado dentro de la interfaz del Marketplace del proyecto TESLA. Esta interfaz es web y está implementada haciendo uso del modelo vista controlador ofrecido por el Framework Spring[18]. De esta forma, el servidor de la aplicación web está implementado mediantes métodos contenidos en clases controladoras de Spring. Estas clases se comunican con la vista y ejecutan la funcionalidad definida en los gestores. La vista de

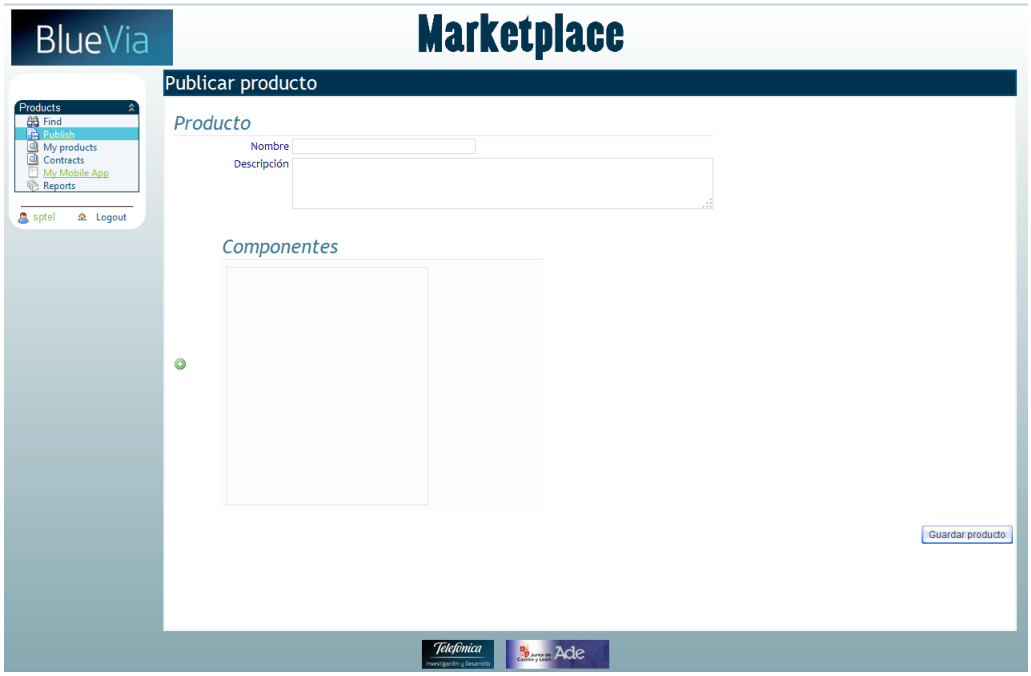

Figura 7.8: Interfaz de definición de producto

la aplicación está implementada mediante ficheros JSP[19] y Javascript. JSP permite mostrar los datos obtenidos de la base de datos o de los modelos XML. Javascript permite implementar interfaces dinámicas que faciliten la creación dinámica de estructuras en forma de árbol como la representada en los ficheros XML.

A continuación se muestran y describen las distintas interfaces implementadas.

7.8.1. Interfaz de definición de producto

La interfaz de definición de producto, que se muestra en la figura 7.8 contiene tres paneles desplegables, dos de los cuales están contenidos dentro del primero de ellos. En la figura 7.8 se puede ver la interfaz con todos los paneles replegados. Existen campos para el nombre y la descripción del producto. En la parte inferior de la interfaz se puede observar un listado de los componentes definidos para el producto. Pueden añadirse nuevos componentes pulsando el botón de agregar. Esto desplegará el panel lateral, como se ve en la figura 7.9. También pueden borrarse componentes seleccionándolos en la lista y pulsando el botón de borrado.

Dentro de este panel, debe asignarse un nombre al componente y pueden añadirse parámetros y unidades. Para ello, debe pulsarse el botón de agregar al lado de cada lista. También pueden borrarse elementos de estas listas.

En la figura 7.10 puede verse la interfaz con el panel de unidades desplegado, donde debe asignarse el nombre del mismo.

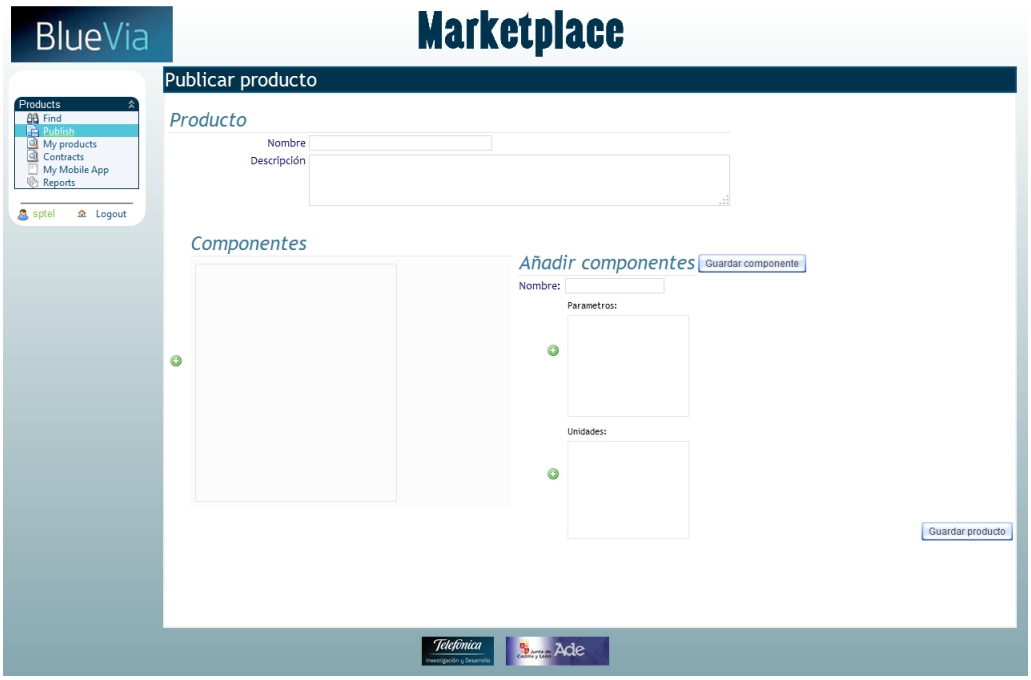

Figura 7.9: Interfaz de definición de producto

En la figura 7.11 puede verse el panel desplegable de definición de un parámetro. En este panel hay campos para el nombre y la unidad. Además, debe seleccionarse el tipo de parámetro entre entero, real y texto. Al seleccionarse el tipo como entero o real, deben seleccionarse el mínimo y el máximo del intervalo. Los campos correspondientes al máximo y el mínimo están restringidos de forma que sólo pueden introducirse caracteres numéricos, y el punto en el caso de tipo real. Al seleccionarse el tipo como texto, el panel cambiará para tomar la forma que se muestra en la figura 7.12. En este caso, deben añadirse los distintos textos en la lista disponible. También es posible eliminar alguno de ellos.

Para poder guardar el producto y pasar a la interfaz de definición del modelo de precio es necesario que ningún campo esté vacío (excepto la descripción), que se defina al menos un componente y que cada componente tenga al menos una unidad. Además, los parámetros de tipo textual deben tener al menos un valor.

7.8.2. Interfaz de definición de modelo de precio

La interfaz de definición de modelo de precio, que se muestra en la figura 7.13 representa un plan de precio. Tras completar la información necesaria en dicha interfaz puede añadirse un nuevo plan de precio o finalizarse la creación del modelo de precio completo.

La parte superior de la interfaz representa la información general del plan de precio. En esta parte se asignan el nombre, el precio de contratación, la duración, los precios máximo y mínimo, los tipos de monedas y países, el precio periódico y la penalización por cancelación del plan.

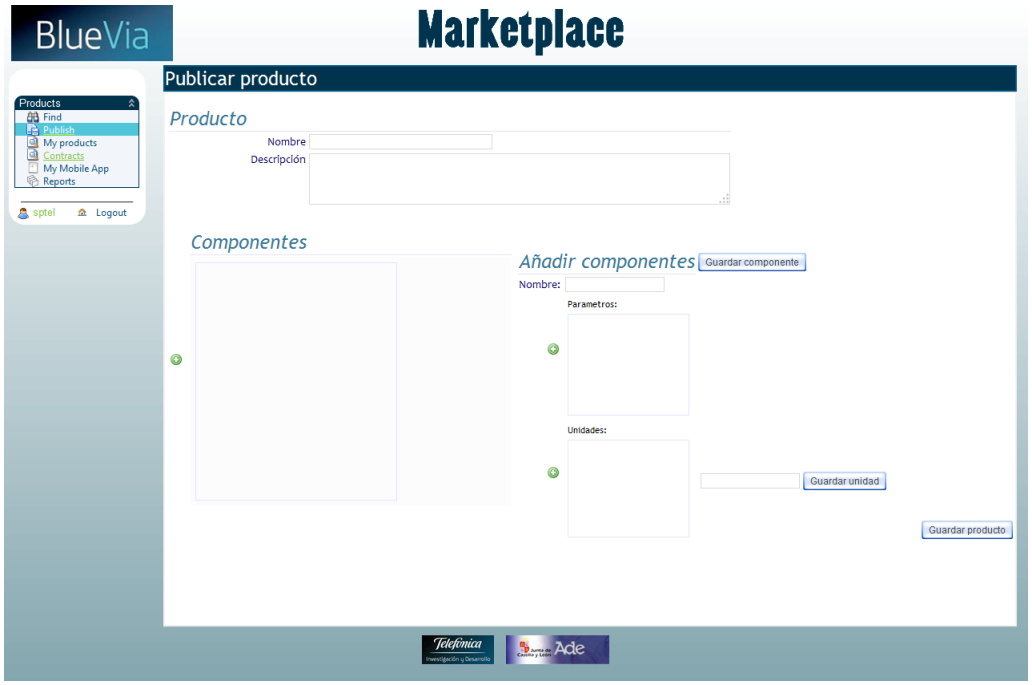

Figura 7.10: Interfaz de definición de producto

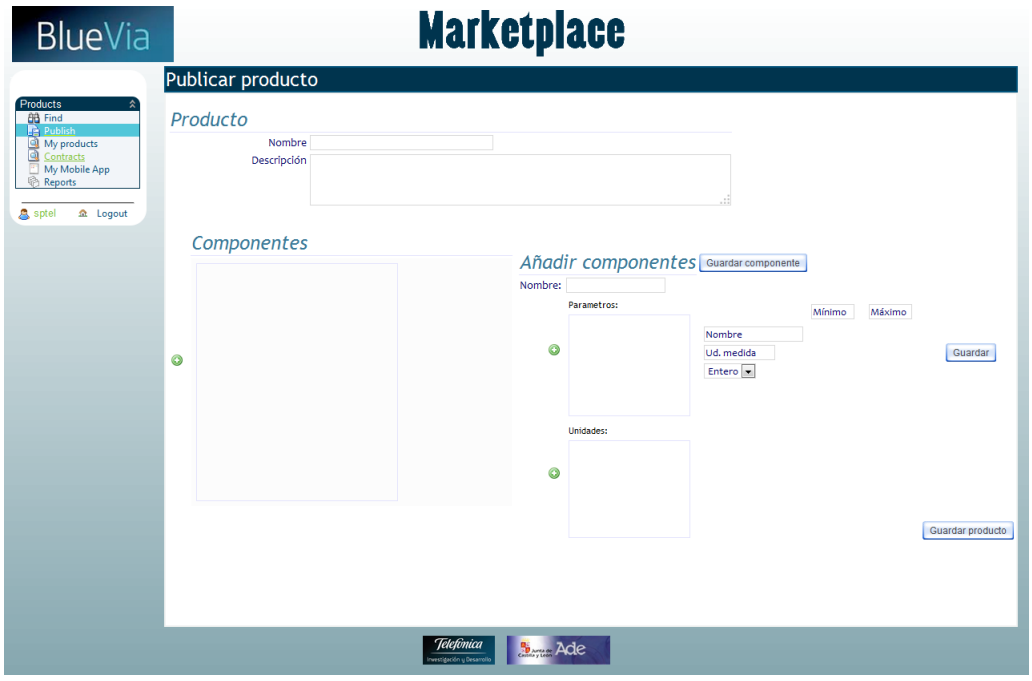

Figura 7.11: Interfaz de definición de producto

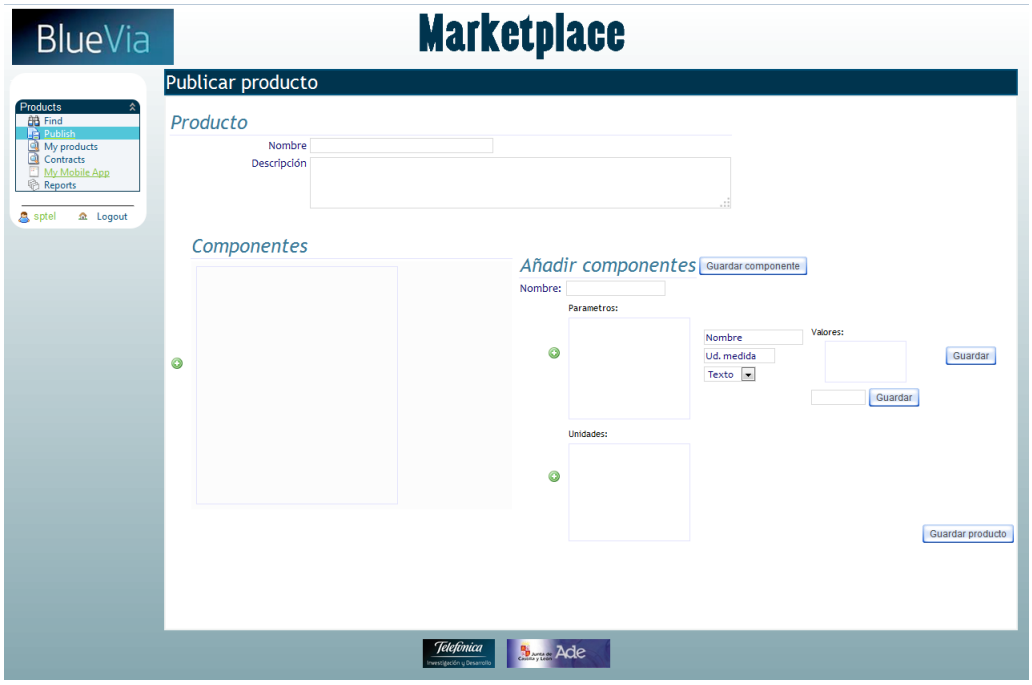

Figura 7.12: Interfaz de definición de producto

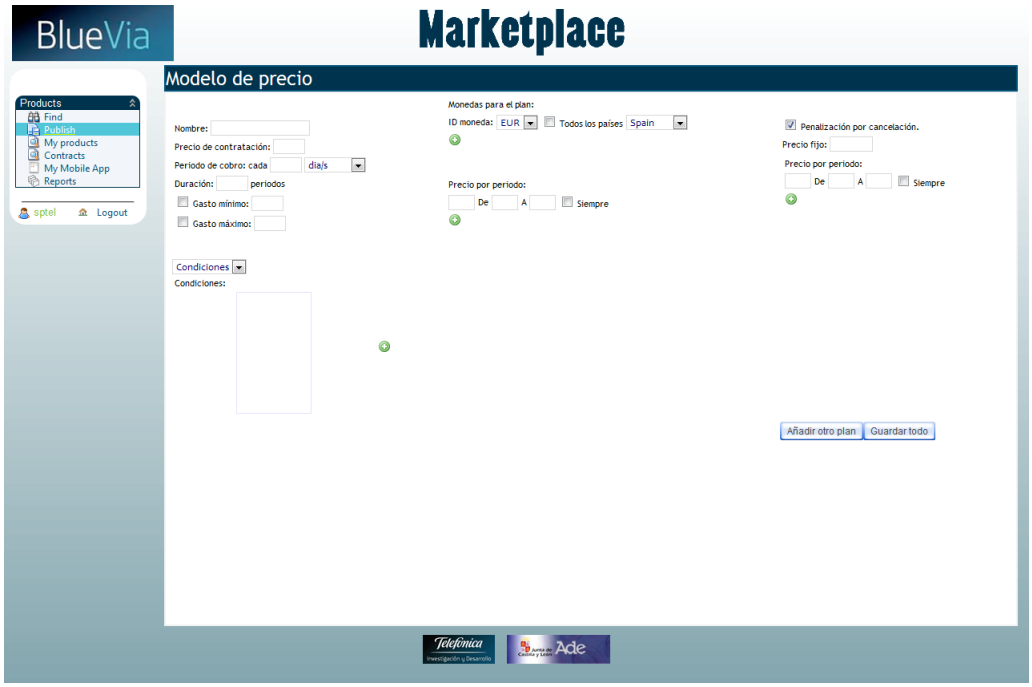

Figura 7.13: Interfaz de definición de precio

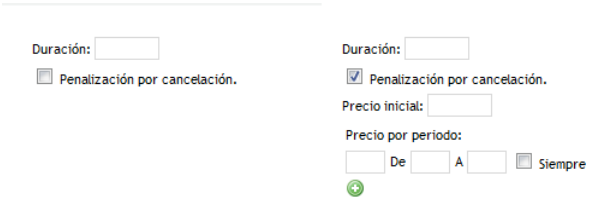

Figura 7.14: Panel de penalización por cancelación temprana

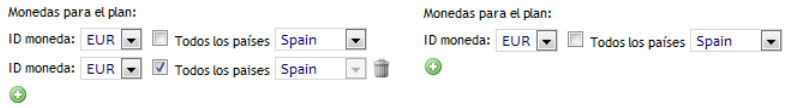

Figura 7.15: Panel de asignación de monedas a un plan

Los precios máximo y mínimo son opcionales y por tanto, sólo es posible asignarles un valor en caso de seleccionar su checkbox asociado.

La penalización por cancelación también es opcional y la información asociada a la misma sólo muestra en caso de seleccionarla, como se muestra en la figura 7.14.

Dado que el plan de precio debe tener al menos una moneda definida, por defecto aparece una. Este plan es válido para el país indicado en el primer desplegable y la moneda indicada en el segundo si no se marca el checkbox de *Todos los países*. En caso de marcarlo, será válido para la moneda indicada independientemente del país. Pueden añadirse más monedas, como se muestra en la figura 7.15. Estas monedas pueden ser distintas a las anteriores o ser iguales y definir un nuevo país. En este caso, esta información se reorganizará dentro de una única moneda en el plan de precio en XML.

El precio por periodo general y el precio por periodo en la cancelación funcionan de forma similar a nivel de interfaz. Debe asignarse como mínimo un precio y puede marcarse la opción *siempre* para indicar que el precio será independiente del periodo. En este caso se ocultará la opción de agregar una nueva línea y se deshabilitarán los extremos del periodo. En caso contrario pueden añadirse o eliminarse distintas líneas de precio que aplican a un periodo determinado. Este periodo se asigna a continuación. Esto puede verse en las figuras 7.16 y 7.17

En la parte inferior de esta interfaz se encuentran las condiciones de precio y las opciones sobre parámetros. Se encuentran en dos paneles que se muestran al seleccionarlos en el menú

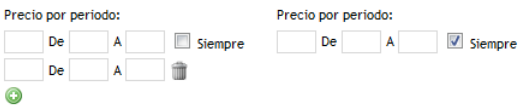

Figura 7.16: Panel de precio por cancelación

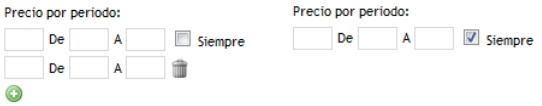

Figura 7.17: Panel de precio periódico

Figura 7.18: Interfaz de asignación de condiciones

desplegable. En el panel de condiciones, que se muestra en la figura 7.18, hay una lista con las condiciones definidas. Para agregar una nueva condición debe seleccionarse el botón de añadir. Esto muestra un panel lateral oculto en el que definir dicha condición.

En la figura 7.19 se muestra este panel, en que hay un selector para el componente y otro para la unidad de medida. La información del segundo selector se actualiza al cambiar el componente en el primero. Además, debe asignarse un precio por unidad a la condición. Para ello hay un cuadro de texto que sólo admite valores numéricos y el punto. Para la selección del tipo de condición hay cinco radiobutton, que representan cada uno de los tipos.

Al seleccionar los tipos de consumo actual y consumo total, se muestran en la parte inferior dos cuadros de texto en los que indicar los extremos del intervalo, como se ve en las figuras 7.20 y 7.21.

Al seleccionar el tipo temporal, en la parte inferior se muestran dos checkbox para indicar si el periodo considera los días de la semana, las horas o ambos como se muestra en la figura 7.22.

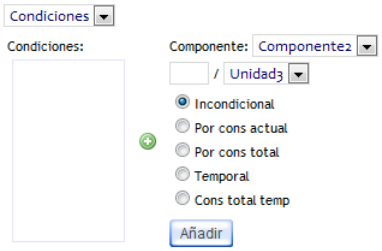

Figura 7.19: Interfaz de asignación de condiciones

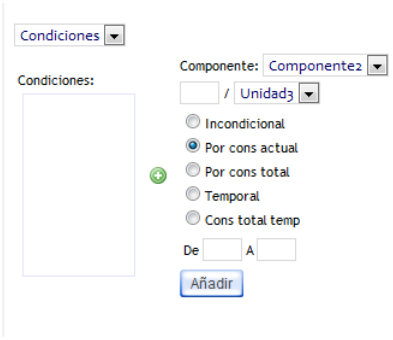

Figura 7.20: Interfaz de asignación de condiciones

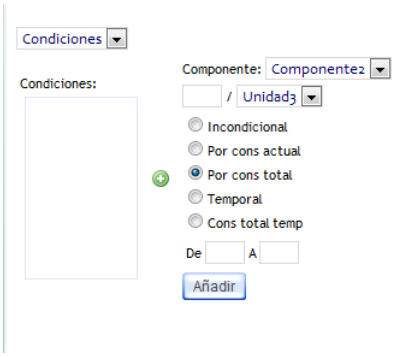

Figura 7.21: Interfaz de asignación de condiciones

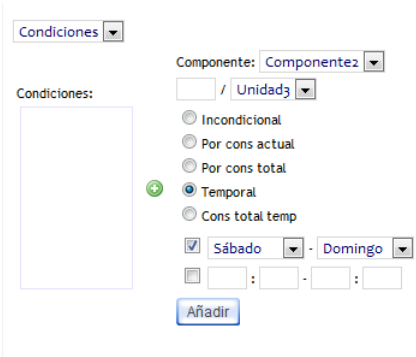

Figura 7.22: Interfaz de asignación de condiciones

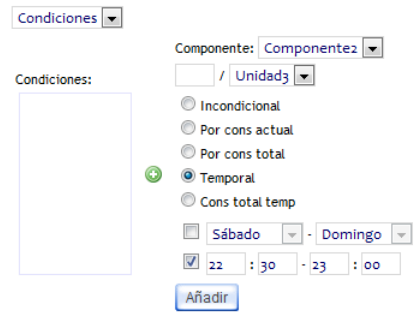

Figura 7.23: Interfaz de asignación de condiciones

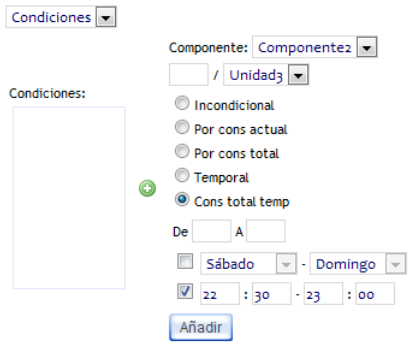

Figura 7.24: Interfaz de asignación de condiciones

Figura 7.25: Interfaz de asignación de opciones

Es obligatorio que uno de los dos checkbox permanezca seleccionado, por lo que si se deseleccionan ambos, uno de ellos vuelve a seleccionarse automáticamente. Cuando estos checkbox están seleccionados, los selectores y cuadros de texto que representan el rango en días y en horas se habilitan para su edición como se ve en la figura 7.23. Los selectores permiten elegir días de la semana definidos en los mismos. Los cuadros de texto horarios permiten definir los rangos en forma de horas y minutos.

La selección del tipo *consumo total temporal* despliega un panel con la información temporal y el rango de valores por consumo total previamente descritos, como se muestra en la figura 7.24.

En el panel de opciones, que se muestra en la figura 7.25, aparece una lista con las opciones definidas.

Para agregar una nueva opción debe pulsarse el botón de añadir. Esto hará visible un panel lateral con la información de la opción, como se muestra en la figura 7.26. Este panel cuenta con dos selectores para el componente y el parámetro dentro del mismo. El selector de parámetro se actualiza en función del selector de componente. Si el parámetro es de tipo numérico, se muestran dos cuadros de texto para introducir los extremos del intervalo de la opción. Si, por el contrario, se trata de un parámetro textual, se muestra un desplegable con los distintos valores posibles, como se muestra en la figura 7.27.

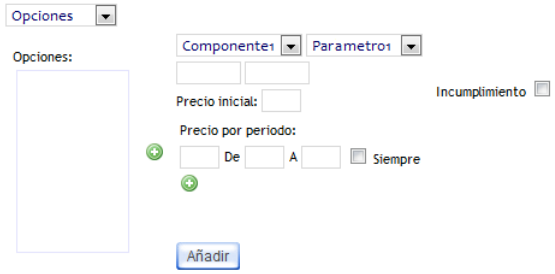

Figura 7.26: Interfaz de asignación de opciones

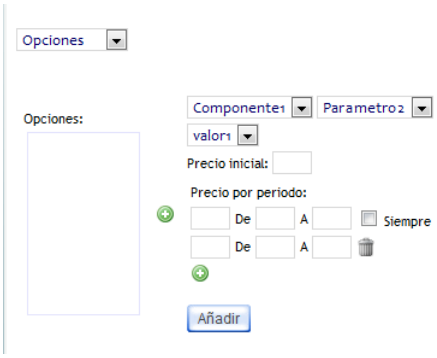

Figura 7.27: Interfaz de asignación de opciones

Cada opción cuenta con un cuadro de texto que sólo admite valores numéricos para el precio inicial. Además, cuenta con una lista de precios por periodo como la descrita anteriormente.

Si el checkbox de penalización por incumplimiento no está marcado, el panel que asigna esta penalización permanece oculto. En caso contrario, este panel se muestra como en la figura 7.28. Este panel cuenta con una lista de cantidades que dependen de un rango, como en el caso del precio por periodo.

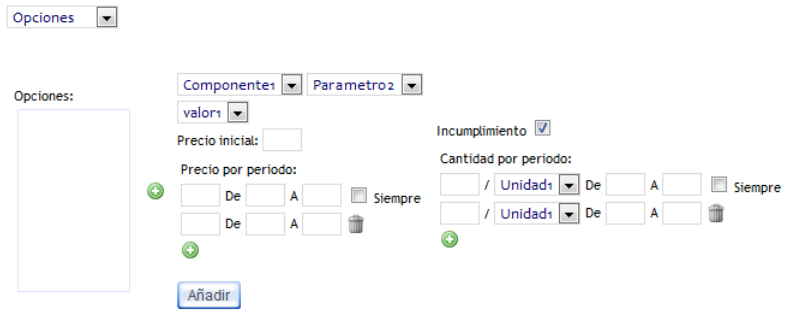

Figura 7.28: Interfaz de asignación de opciones

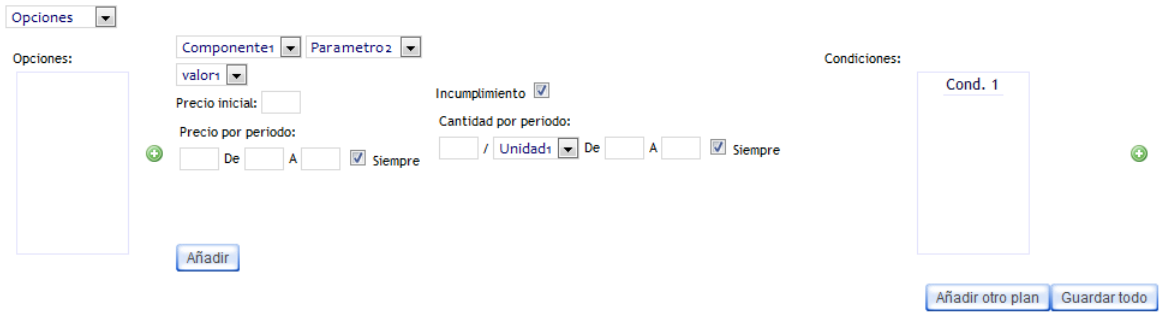

Figura 7.29: Interfaz de asignación de opciones

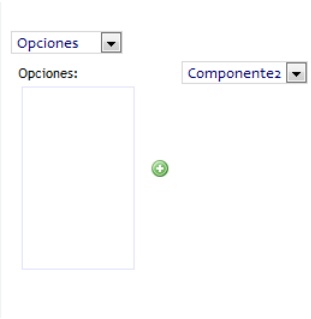

Figura 7.30: Interfaz de asignación de opciones

Además, el panel de opciones permite asignar condiciones dentro de una opción. Esto se realiza con un panel de condiciones idéntico al descrito anteriormente y que se muestra en la figura 7.29.

Si un componente no cuenta con ningún parámetro, el panel se vacía como se muestra en la figura 7.30.

Al guardar una condición o una opción nueva, el panel correspondiente vuelve a su estado inicial, limpiándose todos los valores asignados y reestableciéndose los valores de todos los selectores.

Las comprobaciones y modificaciones dinámicas de esta interfaz se realizan haciendo uso de javascript. La información relativa al producto que en ella se muestra se introduce en la misma mediante el lenguaje de definición de expresiones JSP.

7.8.3. Visualización de productos

La interfaz de visualización de productos, accesible desde el rol de proveedor de servicios y que se muestra en la figura 7.31, permite ver todos los productos definidos por dicho proveedor.

Figura 7.31: Interfaz de visualización de productos

Además, permite borrar cualquiera de estos productos. En este caso, la interfaz se actualiza para no mostrarlo.

La información que muestra esta interfaz se obtiene de la base de datos y se presenta con JSP.

7.8.4. Interfaz de contratación

La interfaz de contratación, que se muestra en la figura 7.32 permite seleccionar un producto para su contratación. Una vez se selecciona un producto y se pulsa el botón de contratación, se muestra una nueva ventana flotante. Esta ventana, que se muestra en las figuras 7.33 y 7.34, muestra la información relativa a cada uno de los planes de precio del producto y a las opciones (si las hubiera) definidas para parámetro del mismo. La selección de un plan habilita la selección de sus opciones. La selección de una opción habilita la selección de sus valores o muestra un cuadro de texto para indicar un valor, según el tipo del parámetro. Además, hay un checkbox para la asignación de límite de gasto. En caso de seleccionarlo, se habilita el cuadro de texto asociado para la asignación de dicho precio.

Todos los cuadros de texto de esta interfaz tienen control para evitar que se introduzcan valores no numéricos.

Esta interfaz está realizada mediante el uso de JSP, obteniéndose la información de los XML de definición de producto y modelo de precio.

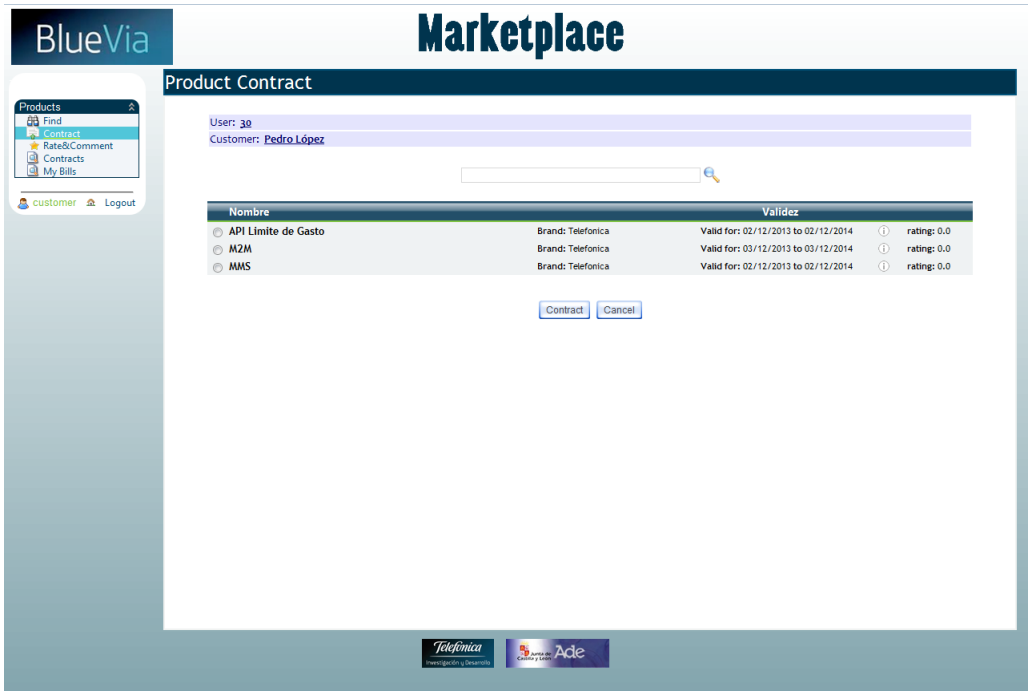

Figura 7.32: Interfaz de contratación

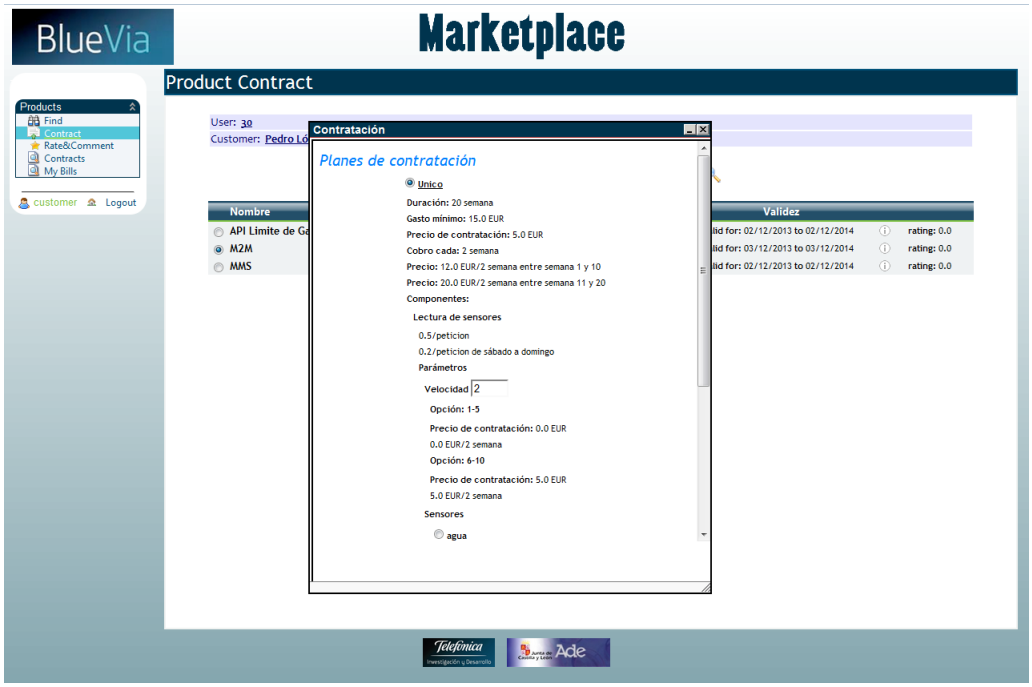

Figura 7.33: Interfaz de contratación

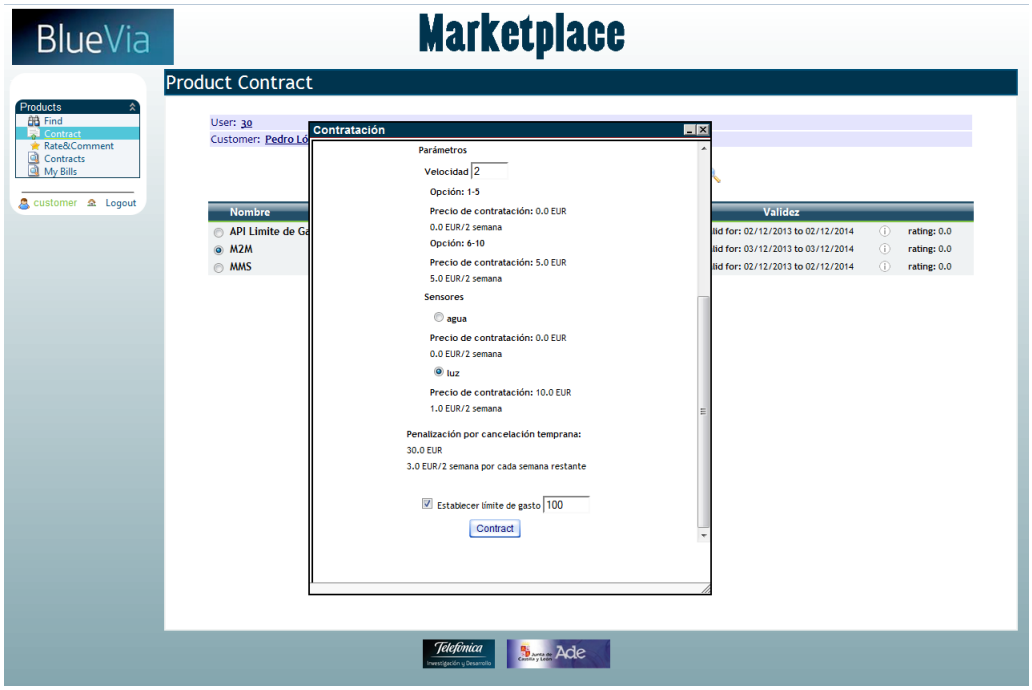

Figura 7.34: Interfaz de contratación

7.8.5. Visualización de contratos

La interfaz de visualización de contratos, accesible en el rol de cliente, muestra una lista de todos los contratos activos que tiene dicho cliente. Esta interfaz, que se muestra en la figura 7.35, permite además, para cada contrato, obtener una factura parcial, que el usuario puede visualizar o descargar, y cancelar el contrato con las consiguientes consecuencias.

7.8.6. Visualización de facturas

La interfaz de visualización de facturas, que se muestra en la figura 7.36, permite a un usuario ver todas las facturas que se han generado para sus contratos, tanto por contratación como periódicas. Aquí no se muestran las facturas manuales, que se destruyen tras su visualización o descarga por parte del usuario. Para la visualización de una factura del listado, el usuario debe descargar la misma. Un ejemplo de factura puede verse en la figura 7.37.

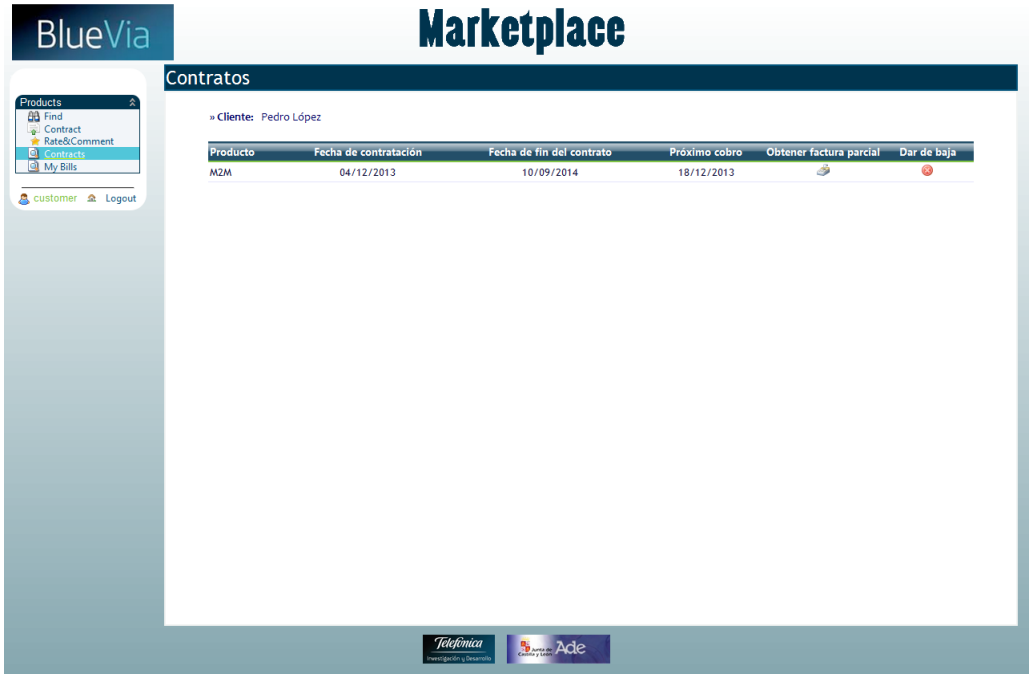

Figura 7.35: Interfaz de visualización de contratos

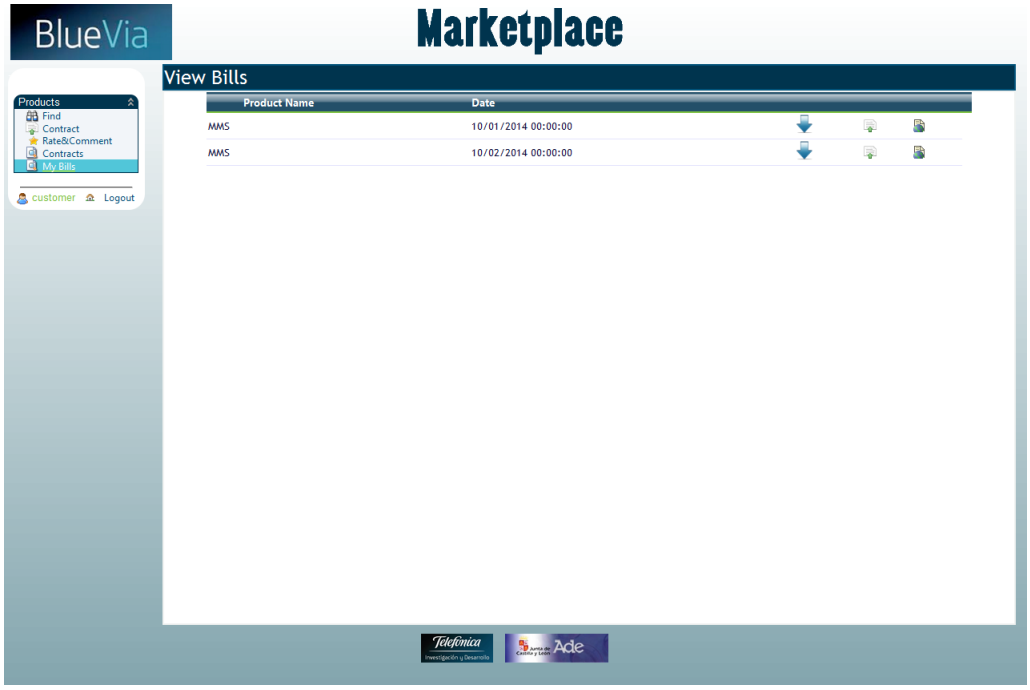

Figura 7.36: Interfaz de visualización de facturas

BlueVia

Factura de TESLA 16/12/2013

CLIENTE: Pedro López

PRODUCTO: Servicio meteorologico

Componente:	Precio:	Descripción	Coste
Consulta humedad	Pay per use time	2013/12/16 07:30:00 Consumo: 10.0 segundos Uso entre 0 y 10	0,00 EUR
Consulta humedad	Pay per use time	2013/12/16 07:30:10 Consumo: 20.0 segundos	2,00 EUR
Consulta humedad	Pay per use quantity	2013/12/16 07:30:01 Consumo: 1 consulta Tarifa 06:00:00-08:00:00	0,05 EUR
Consulta temperatura	Pay per use time	2013/12/16 09:30:00 Consumo: 10.0 segundos Uso entre 0 y 10	0,00 EUR
Consulta temperatura	Pay per use time	2013/12/16 09:30:10 Consumo: 50.0 segundos	5,00 EUR
Consulta temperatura	Pay per use quantity	2013/12/16 09:30:01 Consumo: 1 consulta	0,10 EUR
General	Subscription	Cobro fijo 16/11/2013-16/12/2013	0,00 EUR
General	Subscription	Cobro fijo 16/11/2013-16/12/2013 por Fiabilidad=99	15,00 EUR
General	Subscription	Cobro fijo 16/11/2013-16/12/2013 por Disponibilidad=98.0	10,00 EUR

Total: 32,15 EUR

Figura 7.37: Ejemplo de factura

8

Caso Real

A continuación se va a describir un caso real de uso de la aplicación. En él, se ha definido el producto correspondiente a un servicio de información meteorológica construido sobre M2M (Machine to machine) y un modelo de precio con dos planes para el mismo. Se ha contratado el producto y se han introducido datos de uso para los contratos generados. Se han generado y visualizado facturas y por último se ha cancelado uno de los contratos y se ha borrado el producto.

8.1. Definición del producto

El servicio meteorológico que se ha definido como producto cuenta con dos funcionalidades. La primera de ellas permite consultar la temperatura, mientras que la segunda permite consultar la humedad. Estas dos funcionalidades se definen como dos componentes distintos.

El componente de consulta de temperatura, que se ha definido como se ve en las figuras 8.1 y 8.2, cuenta con un parámetro, que representa la fiabilidad de la información. Este parámetro puede tener dos valores fijos, que son 90 % y 99 % de fiabilidad. El componente tiene dos unidades de medida distintas que son: las consultas y los segundos. De esta forma, se miden tanto el número de consultas como la duración de cada una.

El componente de consulta de humedad, que se define como se ve en las figuras 8.3 y 8.4, cuenta con un parámetro, que representa la disponibilidad de los sensores de información de

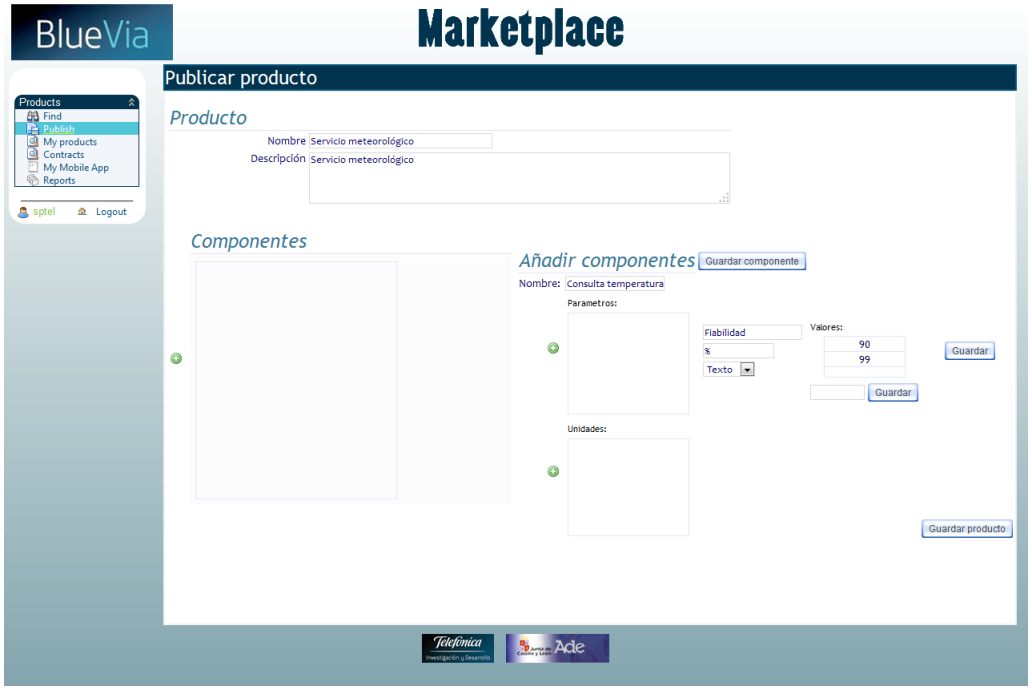

Figura 8.1: Definición del componente de consulta de temperatura

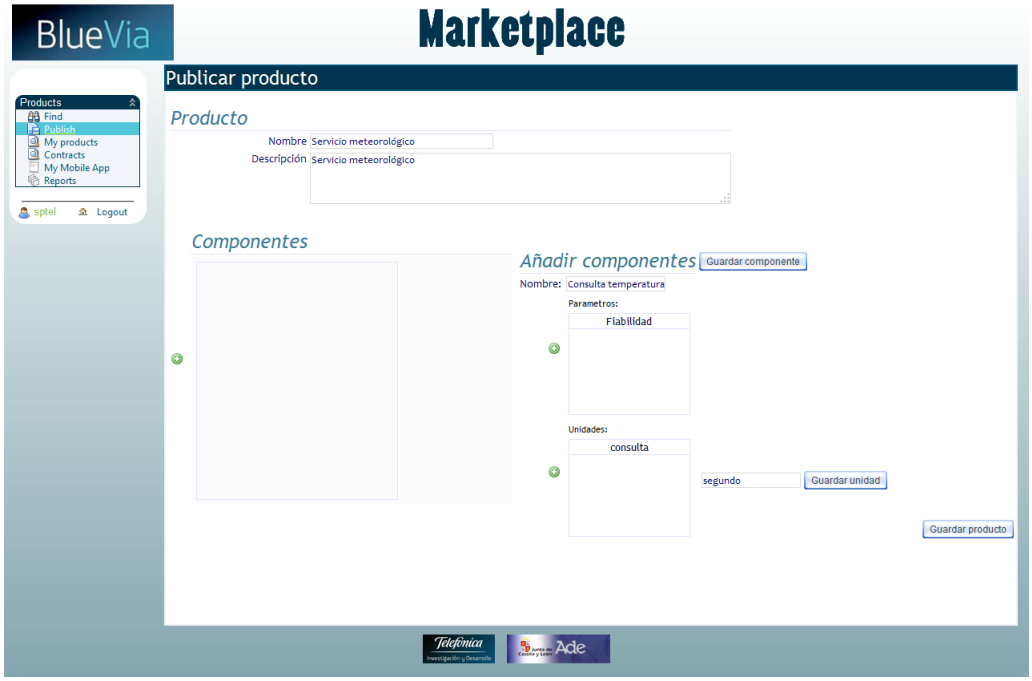

Figura 8.2: Definición del componente de consulta de temperatura

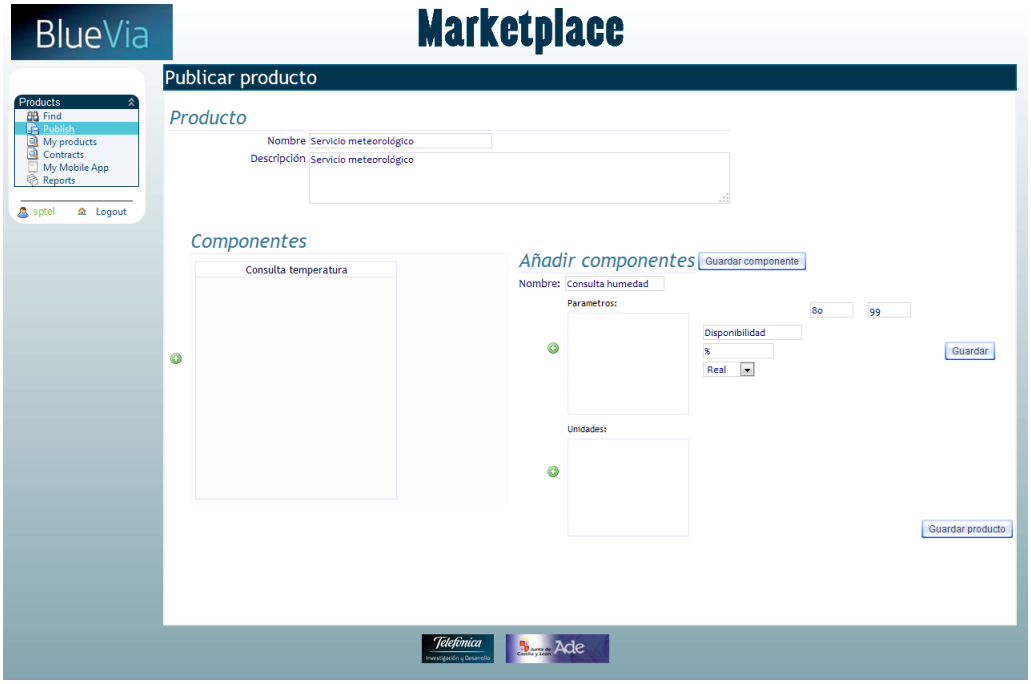

Figura 8.3: Definición del componente de consulta de humedad

humedad. Este parámetro puede tomar cualquier valor real entre 80 y 99. El componente tiene dos unidades de medida distintas que son las consultas y los segundos. De esta forma, se miden tanto el número de consultas como la duración de cada una.

8.2. Definición del modelo de precio

El plan de precio de este producto consta de dos planes: Básico y Premium.

8.2.1. Plan Básico

Características generales

El plan de precio Básico, cuya definición general puede verse en la figura 8.5, tiene un precio de contratación de 10€. La duración de este plan es de un año, con cobro mensual. El precio mensual del mismo es gratuito, aunque debe realizarse un gasto mínimo de 10€. Este plan está disponible para todos aquellos clientes que paguen en euros y para aquellos que paguen en libras esterlinas y residan en el Reino Unido. La cancelación de un contrato sobre este producto no conlleva ninguna penalización económica.

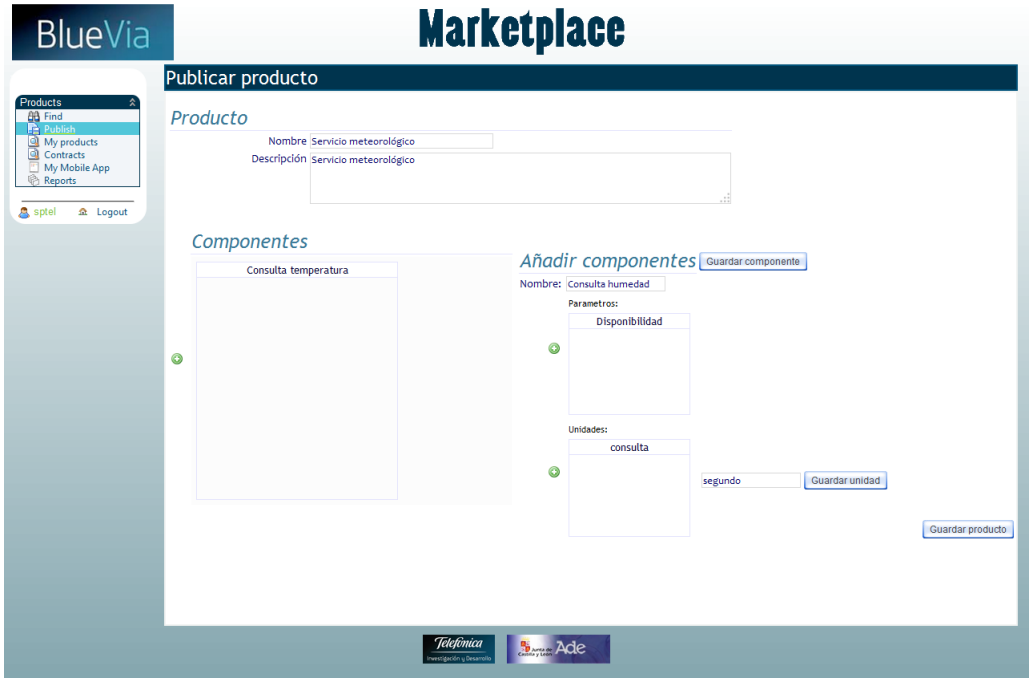

Figura 8.4: Definición del componente de consulta de humedad

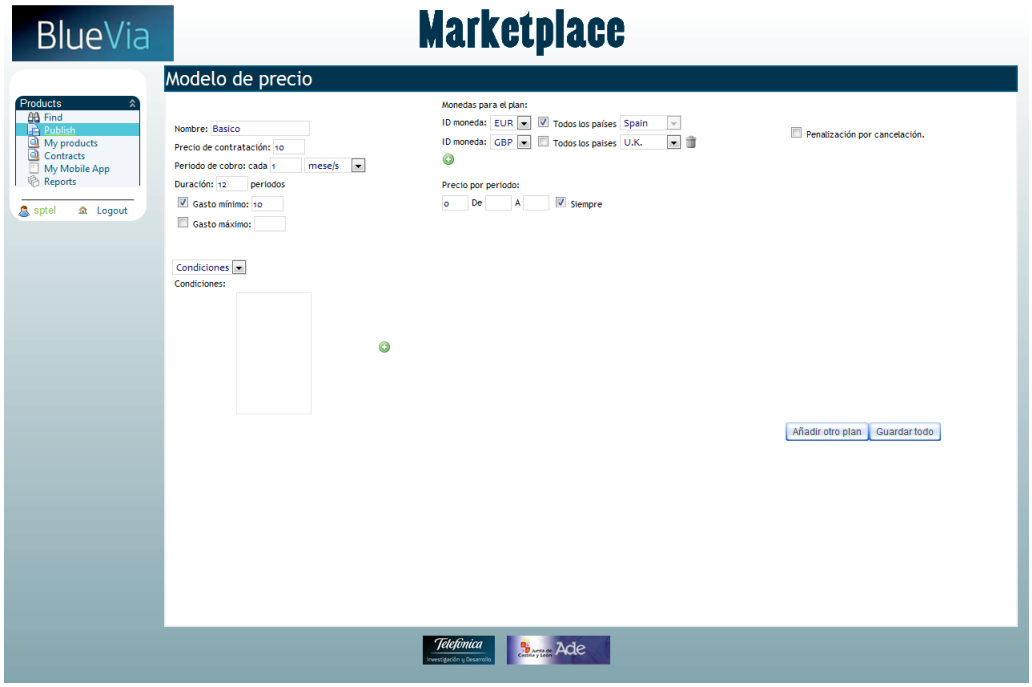

Figura 8.5: Definición general del plan de precio Básico

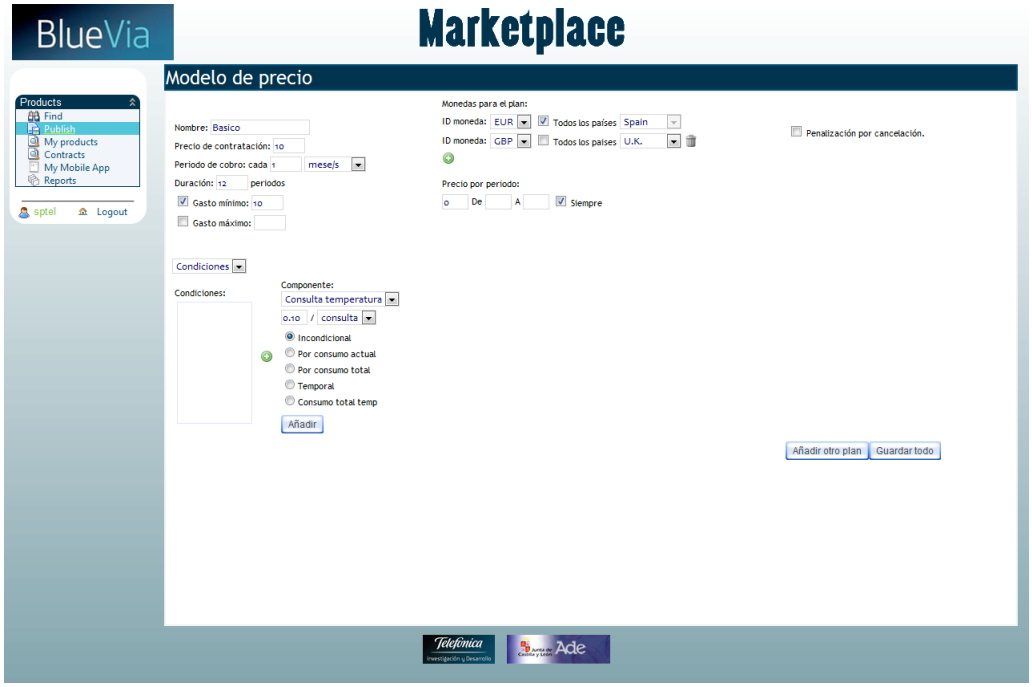

Figura 8.6: Definición de los precios por uso

Precios por uso

Los precios por uso de este producto se definen de la misma forma para los dos componentes (que cuentan con las mismas unidades de medida: consultas y segundos).

- 0,10€/segundo en el caso general.
- 0€/segundo para los 10 primeros segundos de cada uso (10 primeros segundos de cada uso gratuitos).
- 0,10€/consulta en el caso general.
- 0,05€/consulta de 06:00 a 08:00.

Pueden verse algunas de estas definiciones en las figuras 8.6, 8.7 y 8.8.

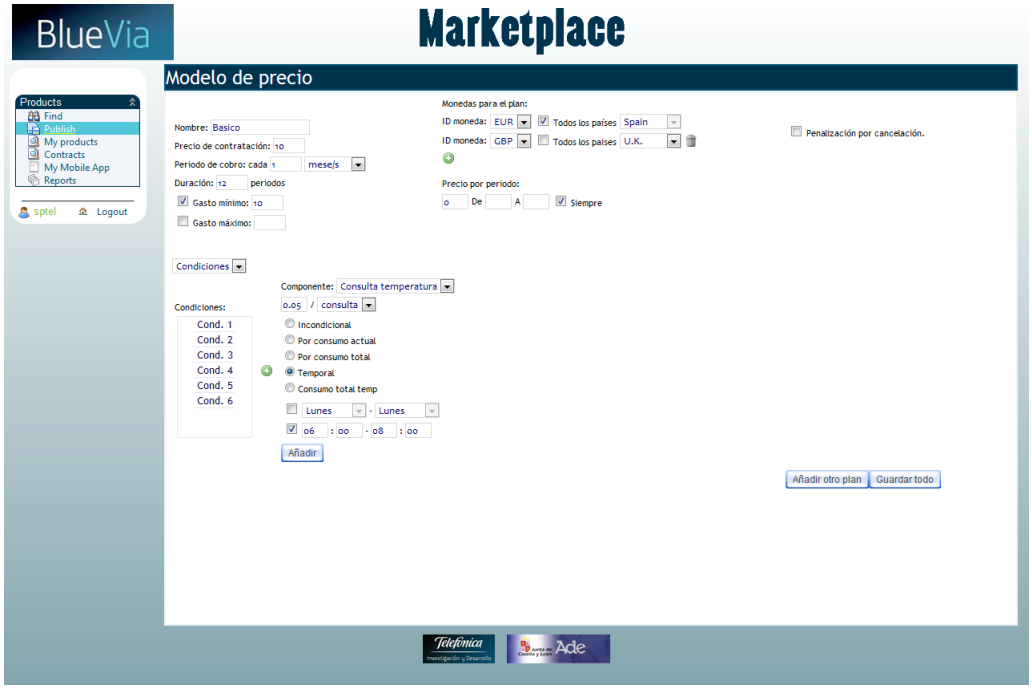

Figura 8.7: Definición de los precios por uso

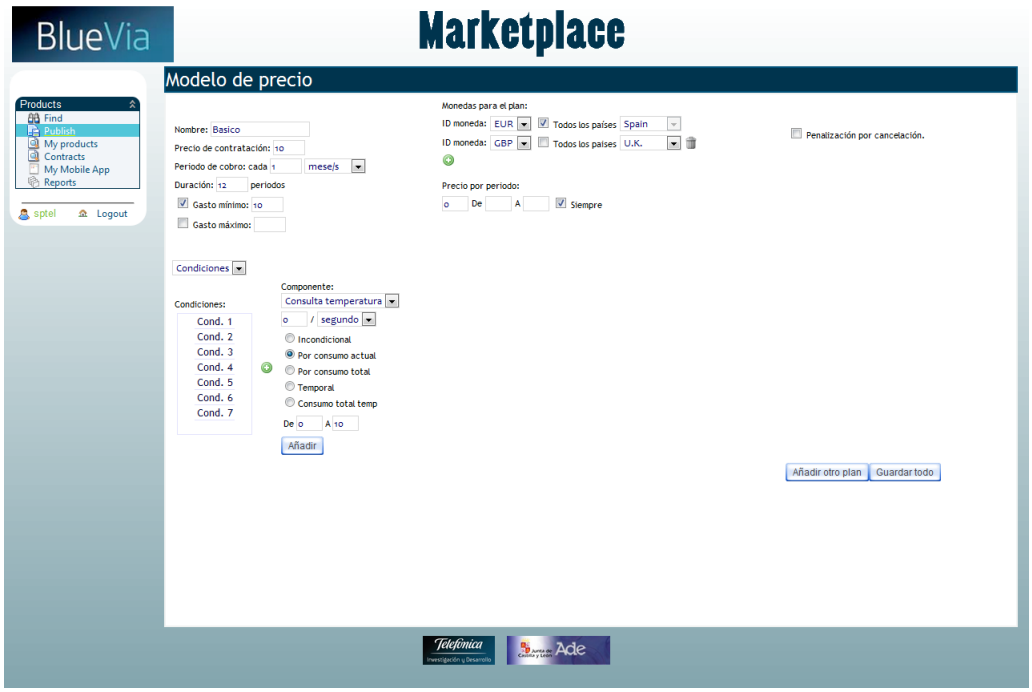

Figura 8.8: Definición de los precios por uso

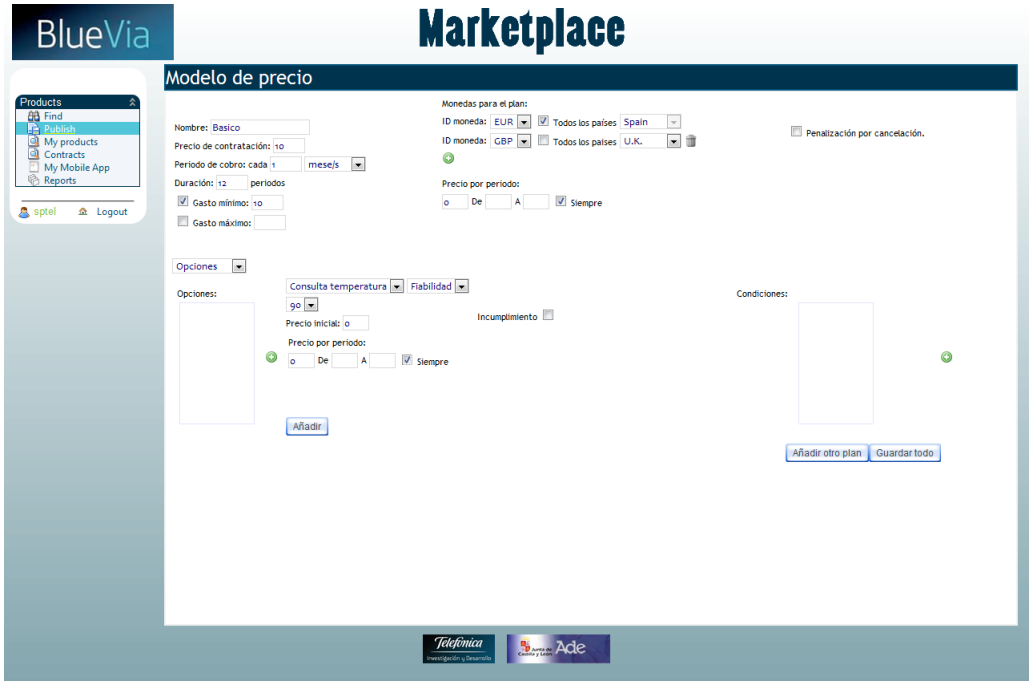

Figura 8.9: Definición de una opción sobre la fiabilidad

Opciones según parámetros

Para cada parámetro que se ha definido (fiabilidad para la consulta de temperatura y disponibilidad para la consulta de humedad), se han definido dos opciones de contratación.

Para el parámetro de fiabilidad, dentro del componente de consulta de temperatura, se han definido dos opciones correspondientes a los valores 90 y 99. La primera de estas opciones, cuya definición puede verse en la figura 8.9, es completamente gratuita, tanto en el precio de contratación como en el precio mensual.

La segunda opción, cuya definición puede verse en la figura 8.10, conlleva un cobro inicial de 20€ y un precio periódico de 15€ para todos los meses. Además, si se produce un incumplimiento en la fiabilidad ofrecida, cada consulta y cada segundo en los que se produzca serán recompensados con una devolución de 0,10€. La selección de estas opciones no conlleva la aplicación de ningún precio por uso nuevo.

Para el parámetro de disponibilidad, dentro del componente de consulta de humedad, se han definido dos opciones correspondientes a los rangos de valores 80-89,9 y 90-99. La primera de estas opciones, cuya definición puede verse en la figura 8.11, es completamente gratuita, tanto en el precio de contratación como en el precio mensual.

La segunda opción, cuya definición puede verse en la figura 8.12, conlleva un cobro inicial de 15€ y un precio periódico de 10€ para todos los meses. La selección de estas opciones no conlleva la aplicación de ningún precio por uso nuevo.

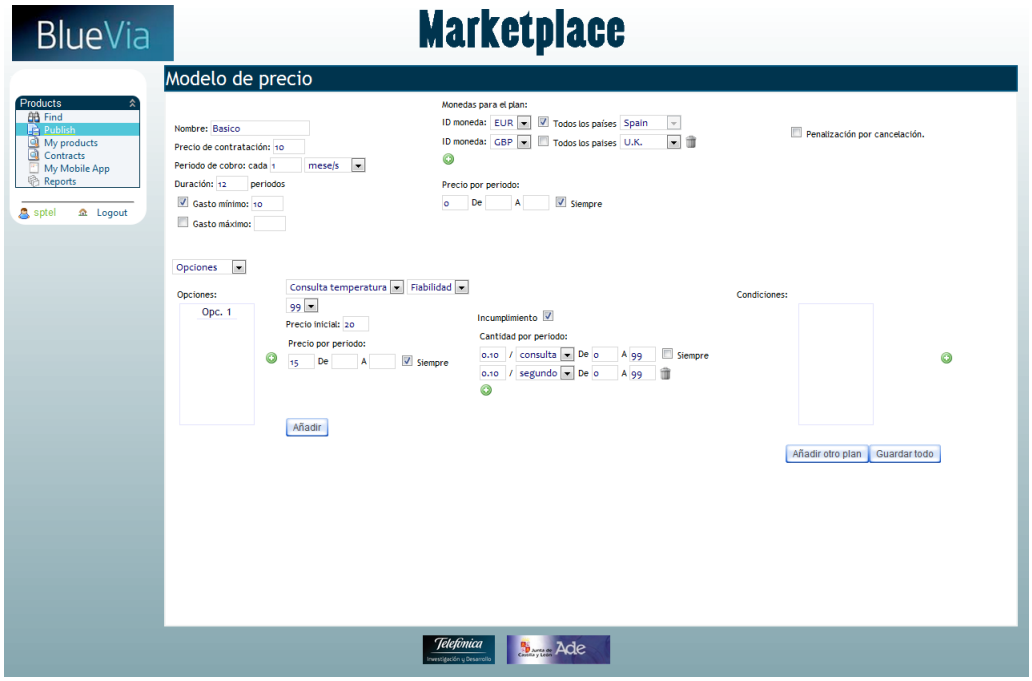

Figura 8.10: Definición de una opción sobre la fiabilidad

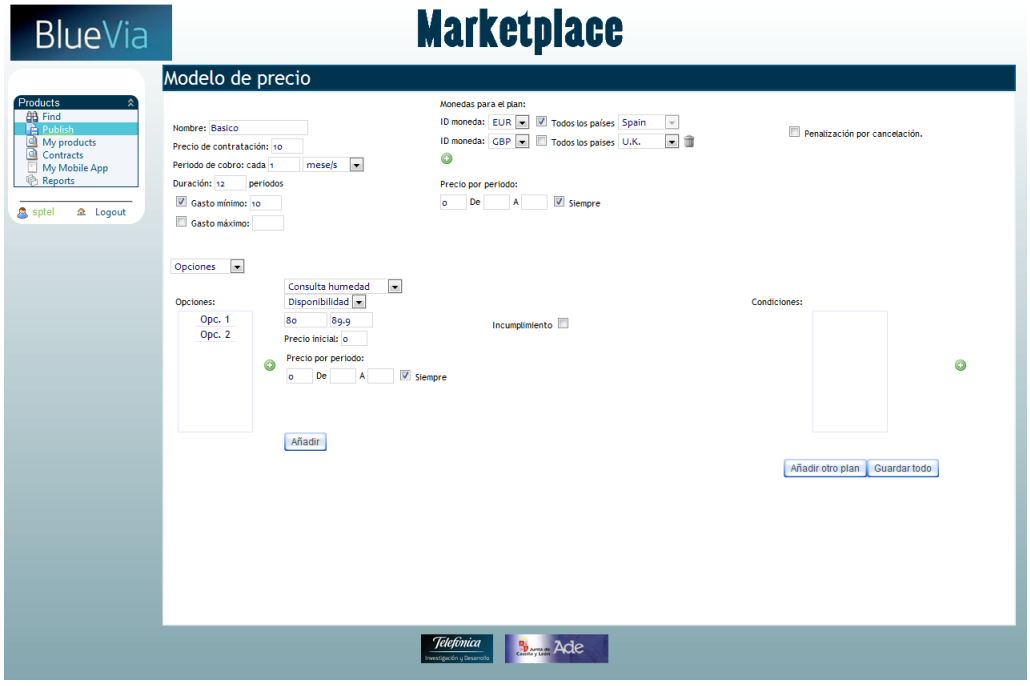

Figura 8.11: Definición de una opción sobre la fiabilidad

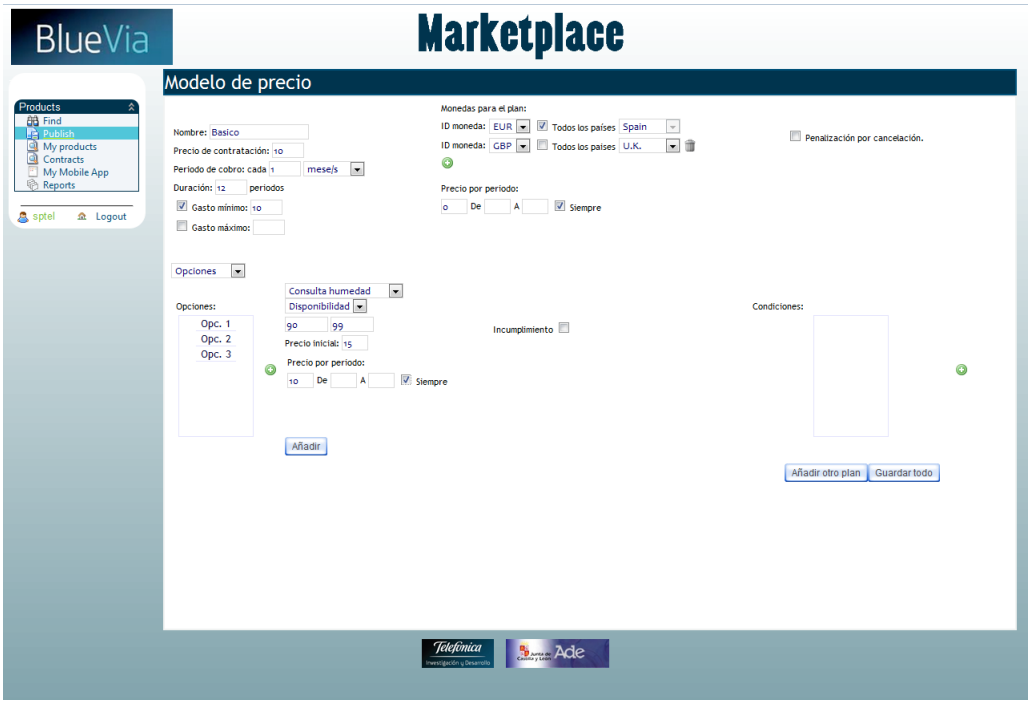

Figura 8.12: Definición de una opción sobre la fiabilidad

8.2.2. Plan Premium

Características generales

El plan de precio Premium, cuya definición general puede verse en la figura 8.13, tiene un precio de contratación de 50€. La duración de este plan es de dos años, con cobro mensual. El precio mensual del mismo es de 10€ durante los 3 primeros meses y de 30€ para el resto. Hay un precio máximo mensual de 150€. Este plan está disponible sólo para aquellos clientes que paguen en euros y residan en España. La cancelación de un contrato sobre este producto con este modelo de precio conlleva un pago de 50€ al que se añaden 5€ por cada mes restante entre los 18 primeros.

Precios por uso

Los precios por uso de este producto se definen de la misma forma para los dos componentes (que cuentan con las mismas unidades de medida, consultas y segundos).

- Consultas gratuitas.
- 0,05€/segundo en el caso general.
- 0€/segundo para los 10 primeros segundos de cada uso (10 primeros segundos de cada uso gratuitos).

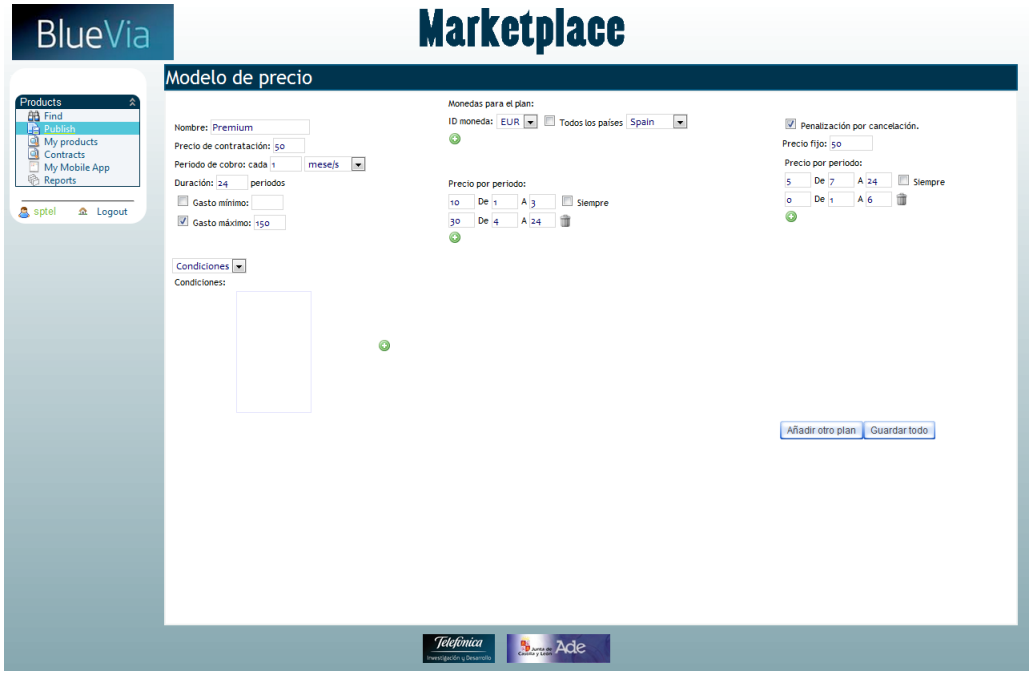

Figura 8.13: Definición general del plan de precio Premium

- 0€/segundo los fines de semana (sábados y domingos).

Pueden verse algunas de estas definiciones en las figuras 8.14, 8.15, 8.16 y 8.17.

Opciones según parámetros

Para el parámetro de fiabilidad, dentro del componente de consulta de temperatura, se ha definido una única opción de valor 99. Esta opción, cuya definición puede verse en la figura 8.18, es gratuita, por lo que este plan Premium incluye una fiabilidad del 99 % gratuita.

Para el parámetro de disponibilidad, dentro del componente de consulta de humedad, se han definido dos opciones correspondientes a los rangos de valores 90-95 y 95-99. La primera de estas opciones, cuya definición puede verse en la figura 8.19, es completamente gratuita, tanto en el precio de contratación como en el precio mensual.

La segunda opción, cuya definición puede verse en la figura 8.20, conlleva un cobro inicial de 10€ y un precio periódico de 5€ para todos los meses. La selección de estas opciones no conlleva la aplicación de ningún precio por uso nuevo.

Figura 8.14: Definición de los precios por uso

Figura 8.15: Definición de los precios por uso

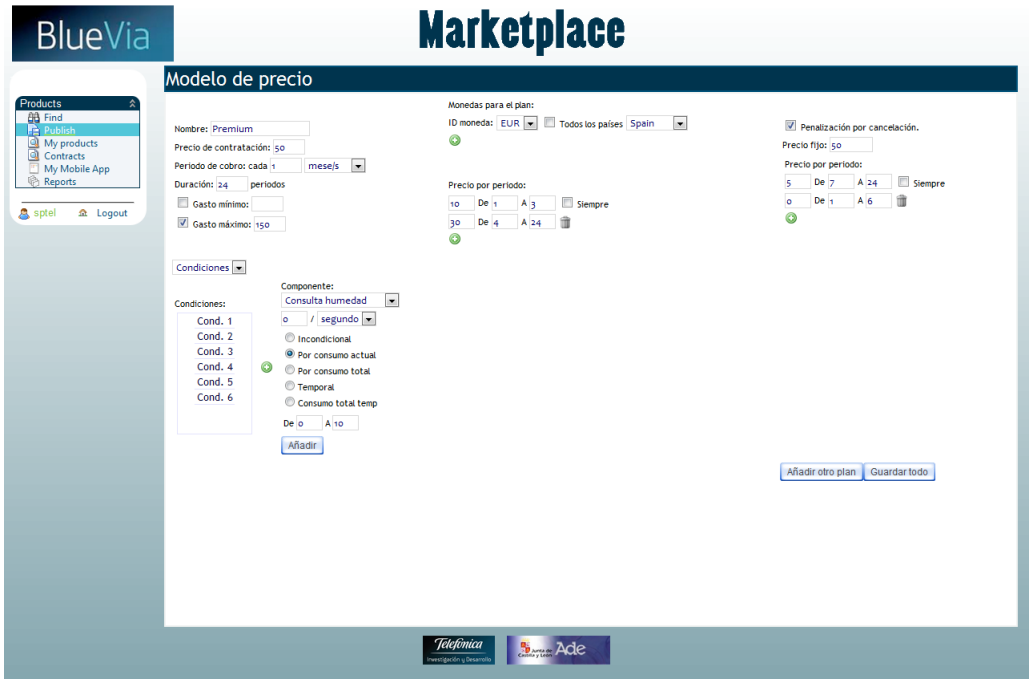

Figura 8.16: Definición de los precios por uso

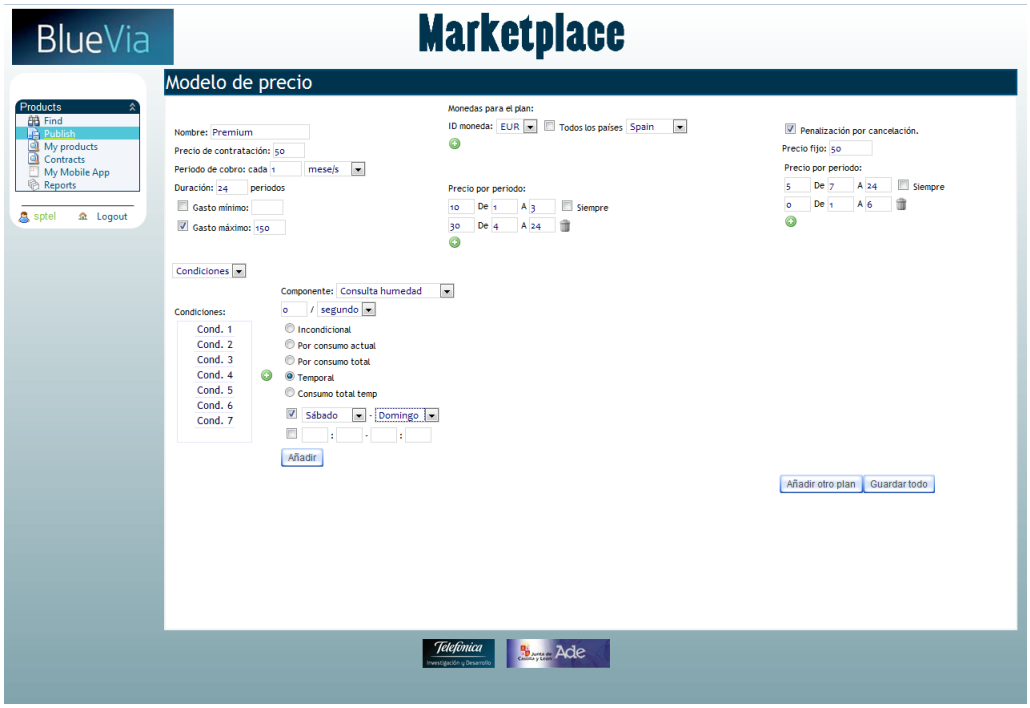

Figura 8.17: Definición de los precios por uso

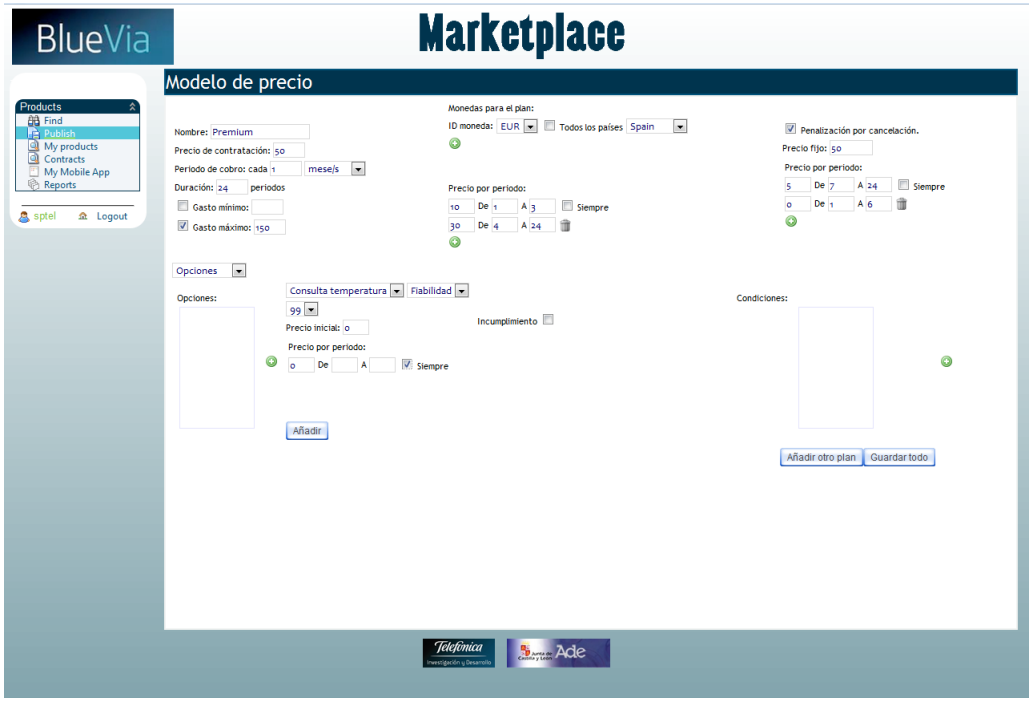

Figura 8.18: Definición de una opción sobre la fiabilidad

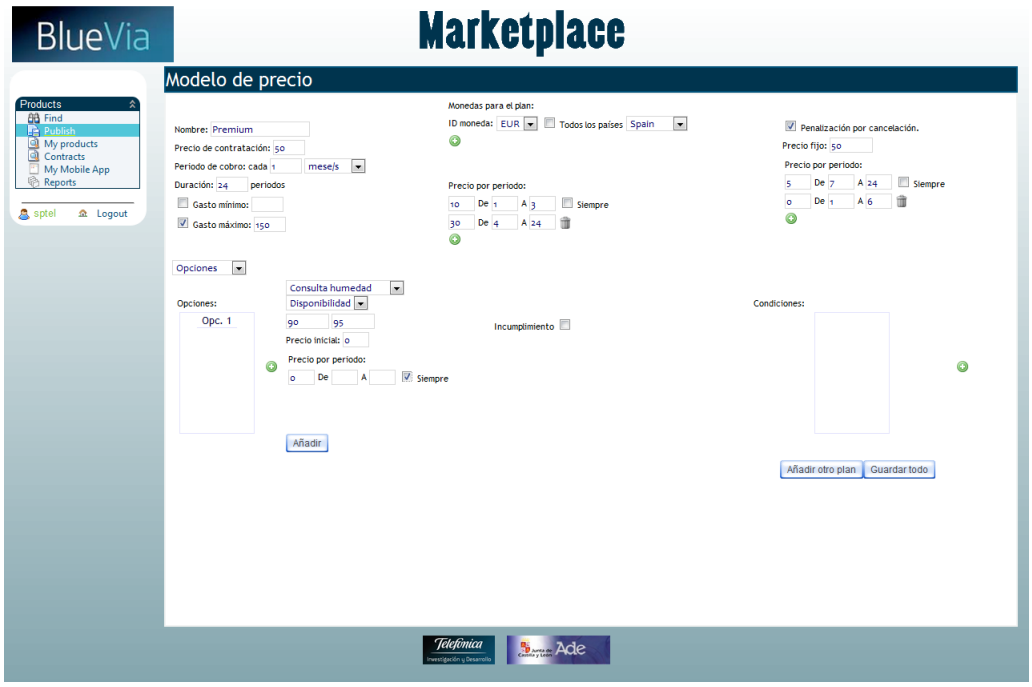

Figura 8.19: Definición de una opción sobre la disponibilidad

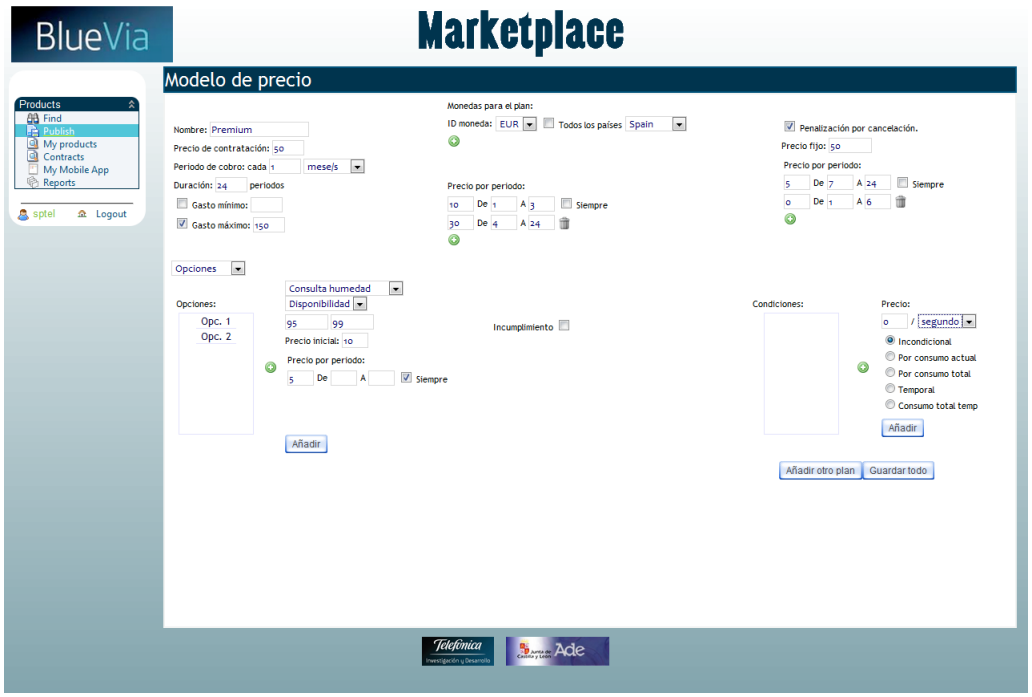

Figura 8.20: Definición de una opción sobre la disponibilidad

8.3. Contratación del producto

A continuación se muestra el proceso de contratación del producto definido. Este producto se ha contratado con dos clientes distintos. Cada uno de ellos lo ha contratado con un plan diferente. Para comenzar el proceso de contratación, en ambos casos, se ha seleccionado el producto deseado de la lista de productos disponibles como se ve en la figura 8.21.

8.3.1. Contratación del plan Básico

En la contratación del plan Básico, que se puede ver en las figuras 8.22, 8.23 y 8.24, se han seleccionado, además del propio plan, una fiabilidad del 99 % para las consultas de temperatura y una disponibilidad del 98 % para las consultas de humedad. Además, se ha establecido un límite de gasto mensual de 60€ tras el cual no se prestará más servicio.

8.3.2. Contratación del plan Premium

En la contratación del plan Premium, que se puede ver en las figuras 8.25 y 8.26, se ha seleccionado automáticamente una fiabilidad del 99 % para las consultas de temperatura. Se ha seleccionado además una disponibilidad del 99 % para las consultas de humedad. No se ha establecido ningún límite de gasto.

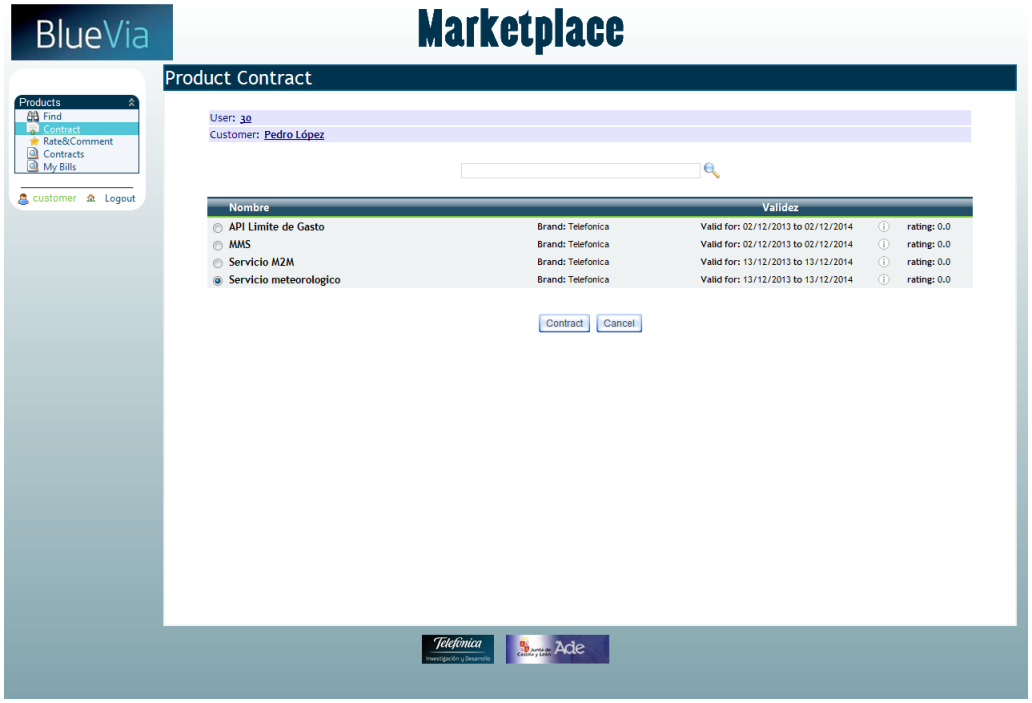

Figura 8.21: Selección de un producto para su contratación

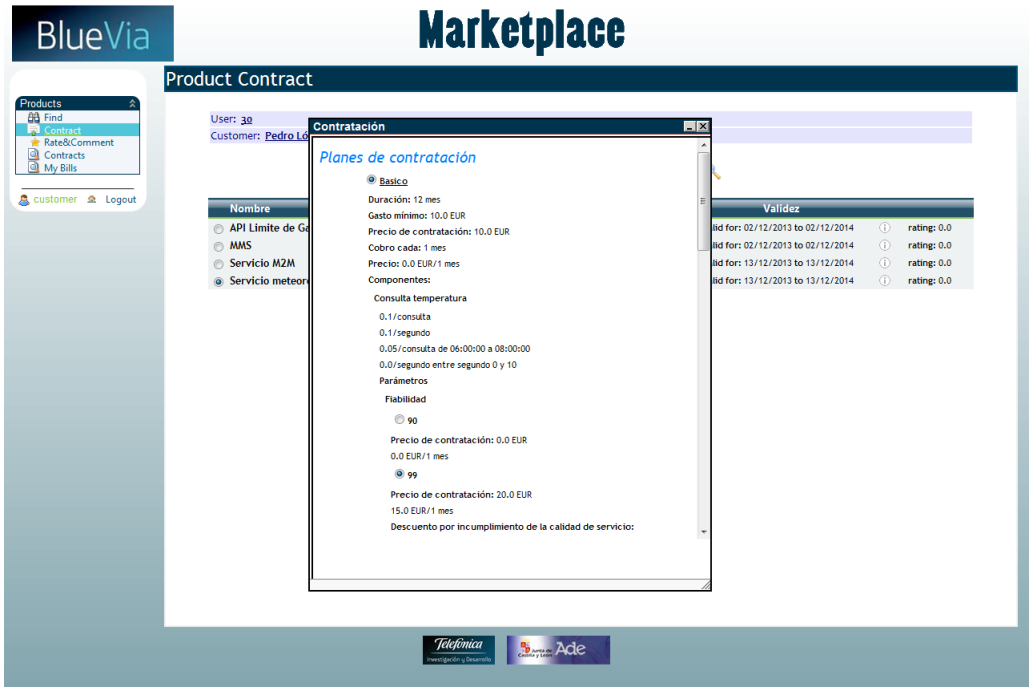

Figura 8.22: Contratación del plan Básico

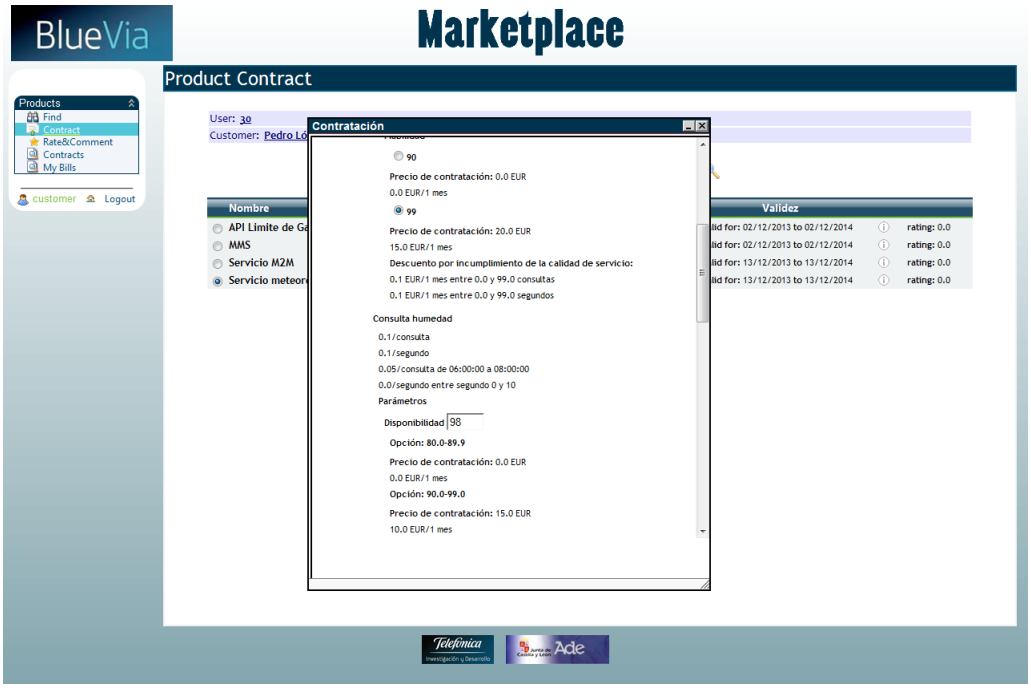

Figura 8.23: Contratación del plan Básico

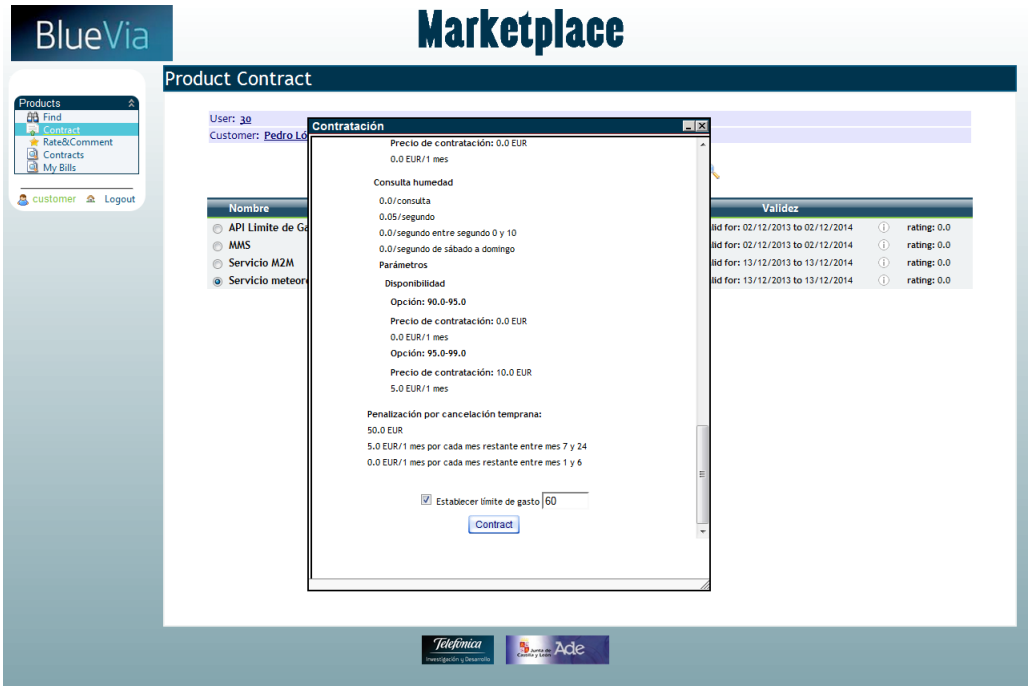

Figura 8.24: Contratación del plan Básico

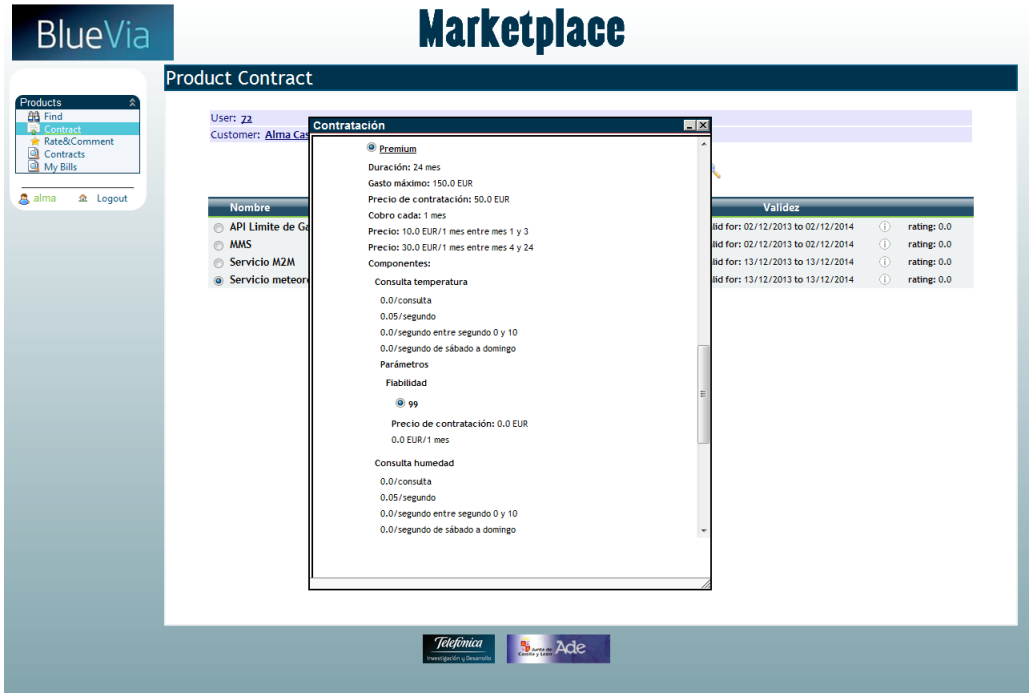

Figura 8.25: Contratación del plan Premium

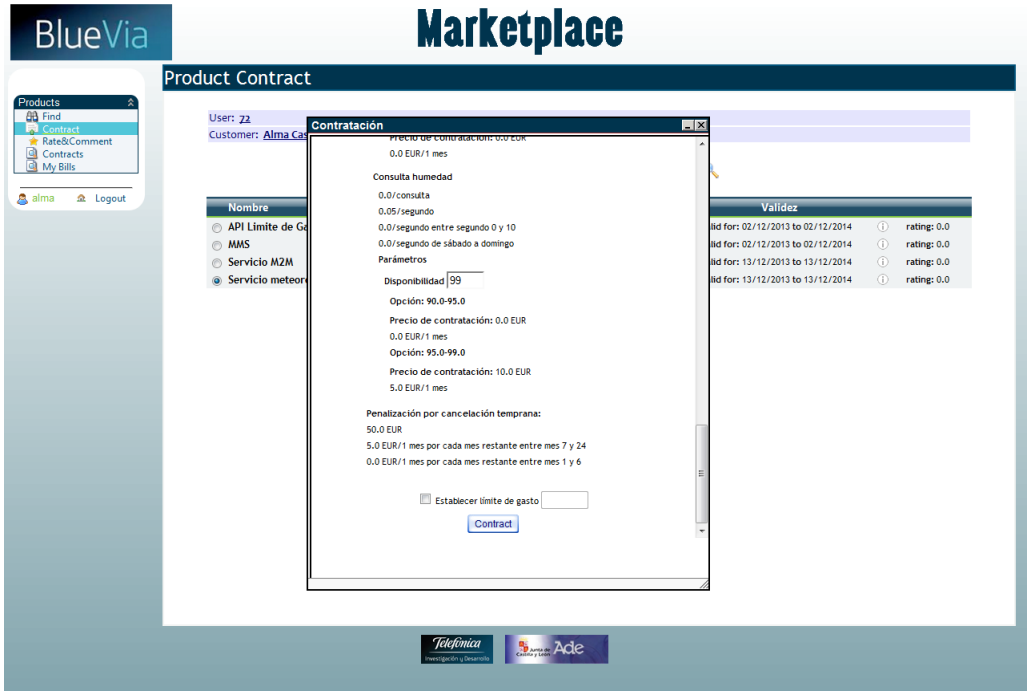

Figura 8.26: Contratación del plan Premium

```
usageDataManager.addQuantityUsage(new UsageCalendar(2013, 11, 16, 9,30, 1), new Long(328), "Consulta temperatura", "consulta", new Long(1));

usageDataManager.addTimeUsage(new UsageCalendar(2013, 11, 16, 9,30, 0), new UsageCalendar(2013, 11, 16, 9,31, 0),
        new Long(328), "Consulta temperatura", "segundo");

usageDataManager.addIncumplimiento(new UsageCalendar(2013, 11, 16, 9,30, 2), new Long(328), "Consulta temperatura", "Fiabilidad", "segundo", new Long(1));

usageDataManager.addQuantityUsage(new UsageCalendar(2013, 11, 16, 7,30, 1), new Long(328), "Consulta humedad", "consulta", new Long(1));

usageDataManager.addTimeUsage(new UsageCalendar(2013, 11, 16, 7,30, 0), new UsageCalendar(2013, 11, 16, 7,30, 30),
        new Long(328), "Consulta humedad", "segundo");


usageDataManager.addQuantityUsage(new UsageCalendar(2013, 11, 17, 10,30, 1), new Long(329), "Consulta temperatura", "consulta", new Long(1));

usageDataManager.addTimeUsage(new UsageCalendar(2013, 11, 17, 10,30, 0), new UsageCalendar(2013, 11, 17, 10,32, 0),
        new Long(329), "Consulta temperatura", "segundo");


usageDataManager.addQuantityUsage(new UsageCalendar(2013, 11, 20, 23,59, 1), new Long(329), "Consulta humedad", "consulta", new Long(1));

usageDataManager.addTimeUsage(new UsageCalendar(2013, 11, 20, 23,59, 0), new UsageCalendar(2013, 11, 21, 00,01, 30),
        new Long(329), "Consulta humedad", "segundo");
```

Figura 8.27: Inserción de datos de uso

8.4. Inserción de datos de uso

Haciendo uso de los métodos para la inserción de datos de uso que se han definido, se han introducido datos de uso para estos contratos, cuyos identificadores son respectivamente 328 y 329.

En la figura 8.27 puede verse el código utilizado para la inserción de los datos de uso.

Para el contrato de plan Básico, los datos introducidos son los siguientes:

- Una consulta de la temperatura con duración de 60 segundos e incumplimiento de la fiabilidad requerida en 1 segundo.
- Una consulta de la humedad con duración de 30 segundos a las 7:30.

Para el contrato del plan Premium, los datos introducidos son:

- Una consulta de la temperatura con duración de 120 segundos.
- Una consulta de la humedad con duración de 150 segundos, de los cuales 90 segundos son un sábado.

8.5. Visualización de facturas y facturación manual

A continuación se han visualizado las facturas de contratación generadas para cada usuario. Para ello, se ha accedido a la interfaz de visualización como se muestra en la figura 8.28 para cada uno de los usuarios.

Se ha descargado la factura correspondiente a la contratación del servicio meteorológico para cada usuario. Estas facturas, que pueden verse en las figuras 8.29 y 8.30, muestran la suma del

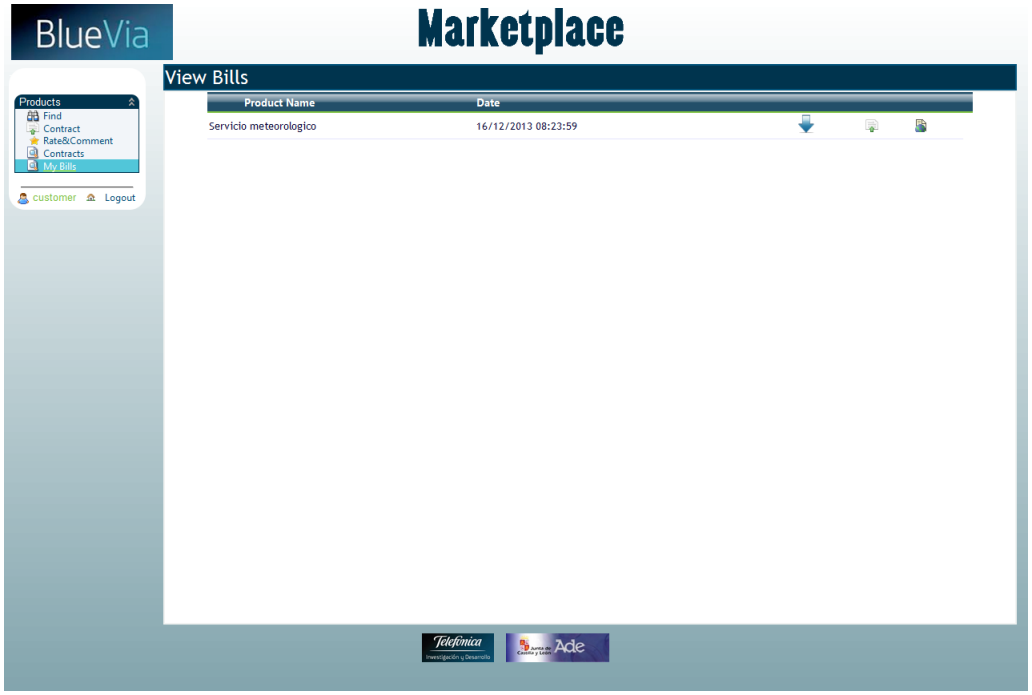

Figura 8.28: Visualización de facturas

Figura 8.29: Factura de contratación para el plan Básico

Figura 8.30: Factura de contratación para el plan Premium

precio de contratación del plan y los precios de contratación de las opciones elegidas para los distintos parámetros.

Se han generado también facturas manuales para estos contratos según los datos de uso introducidos accediendo a la interfaz de visualización de contratos como se muestra en la figura 8.31.

Las facturas obtenidas muestran la tarificación de los datos de uso introducidos. El cálculo del cobro final del periodo se actualiza al finalizar el periodo de cobro y realizarse la facturación automática. En la figura 8.32 puede verse la factura correspondiente al contrato del plan Básico. En la figura 8.33 puede verse la factura correspondiente al contrato del plan Premium.

8.6. Cancelación del contrato

Se ha cancelado uno de los contratos realizados desde la interfaz de visualización de contratos, como se muestra en la figura 8.34. El cálculo de la penalización por cancelación se mostrará en la factura final del mismo.

8.7. Borrado del producto

Para realizar el borrado del producto, se ha accedido a la interfaz de visualización de productos que se muestra en la figura 8.35 desde el rol de proveedor de servicios que lo ha definido. Se ha borrado el producto, finalizando la validez del mismo y cancelándose todos los contratos asociados aún vigentes, sin penalización económica para los clientes que los habían realizado.

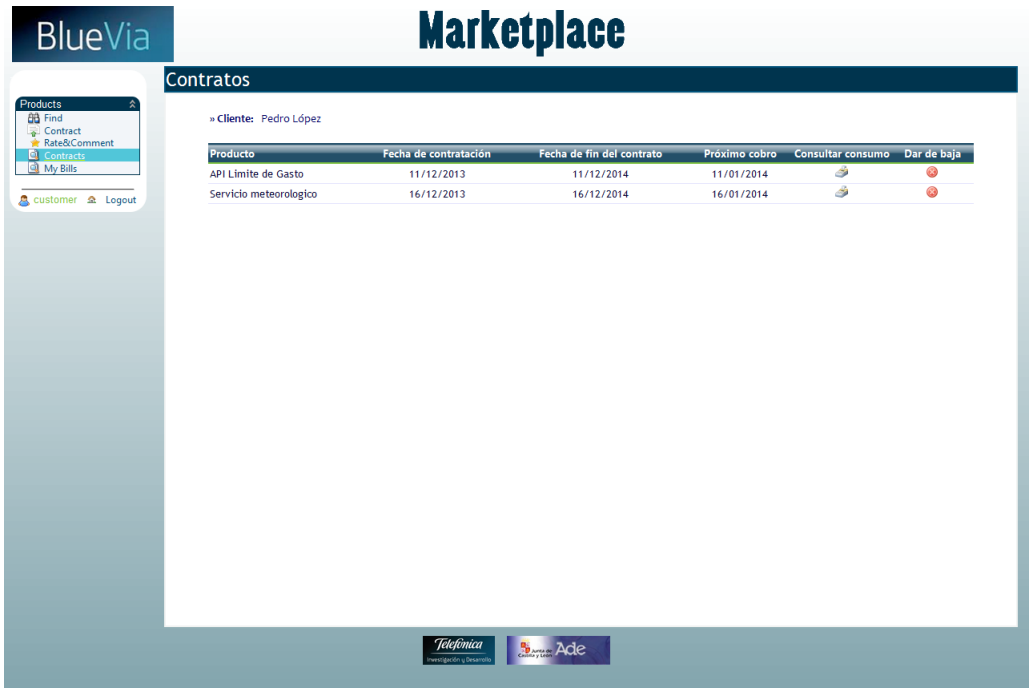

Figura 8.31: Visualización de contratos

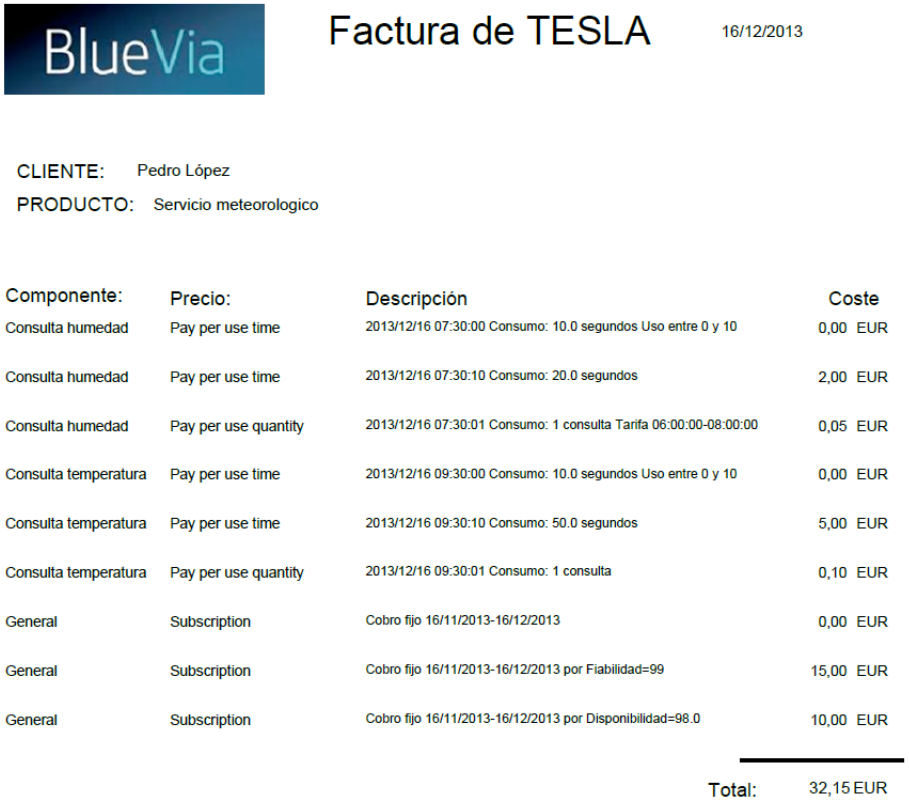

Figura 8.32: Factura manual para el plan Básico

BlueVia

Factura de TESLA 16/12/2013

CLIENTE: Alma Castillo
PRODUCTO: Servicio meteorologico

Componente:	Precio:	Descripción	Coste
Consulta humedad	Pay per use time	2013/12/20 23:59:00 Consumo: 10.0 segundos Uso entre 0 y 10	0,00 EUR
Consulta humedad	Pay per use time	2013/12/20 23:59:10 Consumo: 50.0 segundos	2,50 EUR
Consulta humedad	Pay per use time	2013/12/21 00:00:00 Consumo: 90.0 segundos Tarifa sábado-domingo	0,00 EUR
Consulta humedad	Pay per use quantity	2013/12/20 23:59:01 Consumo: 1 consulta	0,00 EUR
Consulta temperatura	Pay per use time	2013/12/17 10:30:00 Consumo: 10.0 segundos Uso entre 0 y 10	0,00 EUR
Consulta temperatura	Pay per use time	2013/12/17 10:30:10 Consumo: 110.0 segundos	5,50 EUR
Consulta temperatura	Pay per use quantity	2013/12/17 10:30:01 Consumo: 1 consulta	0,00 EUR
General	Subscription	Cobro fijo 16/11/2013-16/12/2013	0,00 EUR
General	Subscription	Cobro fijo 16/11/2013-16/12/2013 por Fiabilidad=99	0,00 EUR
General	Subscription	Cobro fijo 16/11/2013-16/12/2013 por Disponibilidad=99.0	5,00 EUR
		Total:	13,00 EUR

Figura 8.33: Factura manual para el plan Premium

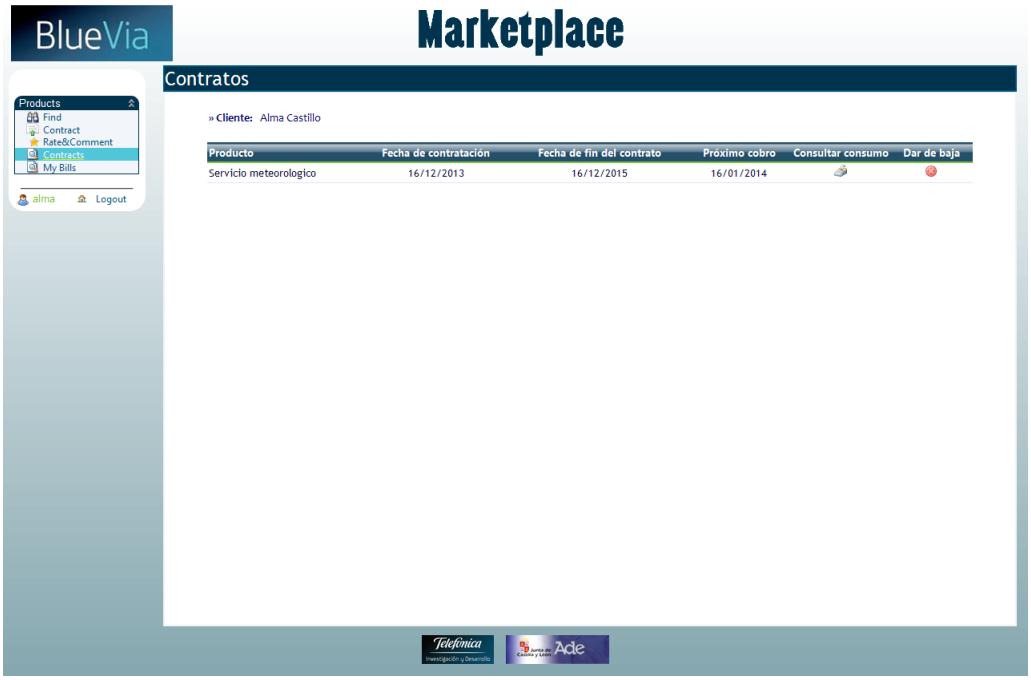

Figura 8.34: Visualización de contratos

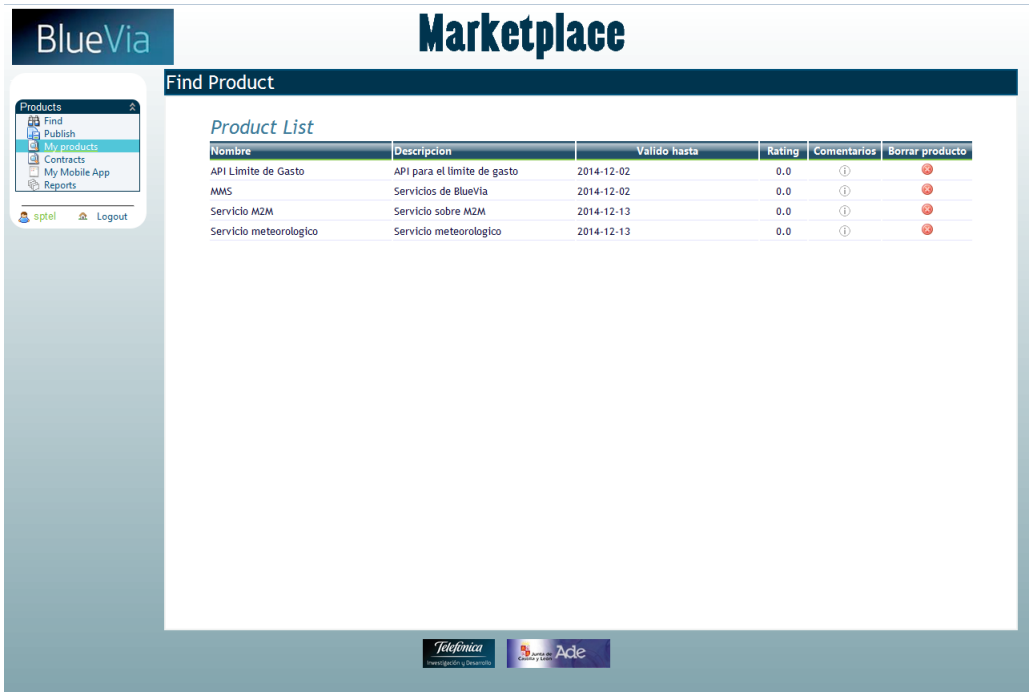

Figura 8.35: Visualización de productos

9

Verificación y validación del sistema

9.1. Verificación

El objetivo de la verificación de esta aplicación es comprobar si la funcionalidad de la misma ha sido implementada correctamente. Con ello, nos referimos a todos los elementos de la misma desde la conexión con la base de datos hasta la interfaz de usuario. Para ello, se han realizado pruebas unitarias y de integración usando distintos métodos y técnicas como se describirá más adelante. Además, se han diseñado pruebas de regresión que permitan verificar nuevamente el sistema tras modificaciones o ampliaciones del mismo. El objetivo de cada una de las pruebas es encontrar el mayor número de errores. Para ello, se ha establecido un plan de pruebas en el que se especifica la actuación esperada del sistema para cada una de ellas.

9.1.1. Estrategia de pruebas

Dada la distinta naturaleza de los elementos que componen la aplicación, la estrategia de pruebas que se ha seguido no es la misma para todos ellos. A continuación se detallan las técnicas utilizadas para la realización de las pruebas de verificación así como qué componentes han sido probados mediante estas técnicas. Para todas las pruebas, se ha diseñado un plan con distintos casos, haciendo énfasis en los casos límite con el objetivo de maximizar el porcentaje de errores encontrados. Para todos los casos de prueba, se ha definido la salida esperada previamente a la realización de la prueba.

Inspección del código

La técnica de inspección de código, consistente en una lectura del mismo con el fin de localizar errores, se ha utilizado para los algoritmos clave de esta aplicación. Estos algoritmos son el algoritmo de particionamiento, explicado en el epígrafe 7.4 y el algoritmo de facturación, explicado en el epígrafe 7.6.

Al tratarse de algoritmos específicos que hacen un tratamiento detallado de los datos, es frecuente la existencia de pequeños errores que causan resultados incorrectos. Dada la longitud del código de estos algoritmos, el número de errores de este tipo suele ser grande, y la combinación de los mismos puede resultar en errores globales difíciles de detectar a través de pruebas unitarias. Por ello, la lectura del código facilita la detección de estos errores y su sencilla corrección, reduciendo en gran medida el número de errores de cada uno de estos algoritmos.

Pruebas unitarias de caja blanca

Las pruebas de caja blanca son aquellas que se ejecutan teniendo en cuenta la estructura interna del programa. Estas pruebas tienen como objetivo la ejecución de todos los caminos existentes en el código, tanto con valores aleatorios como con valores límite. En este caso, se han realizado pruebas de caja blanca sobre los algoritmos antes indicados, que dada su complejidad pueden tener múltiples errores pequeños, y sobre los algoritmos de comprobación de coherencia, que también presentan múltiples caminos.

Además, se han realizado pruebas de caja blanca sobre el componente de lectura de los datos de uso con el fin de minimizar el número de errores que se introduzcan a la aplicación por esta vía, que es el origen del proceso de facturación.

Pruebas unitarias de caja negra

Las pruebas unitarias de caja negra se han realizado para todos los componentes de la aplicación (conexión con la base de datos, gestores, algoritmos de comprobación de coherencia, algoritmo de particionamiento, algoritmo de facturación y lectura de datos de uso). Cada una de las pruebas ha probado un único método Java. Para ello, se ha hecho uso de la herramienta JUnit [25] para la automatización de pruebas unitarias en Java.

Esta herramienta permite definir casos de prueba junto con sus salidas esperadas en lenguaje Java. De forma automática, es posible ejecutar todas las pruebas y obtener un informe de errores de las mismas. Las pruebas unitarias tienen como objetivo detectar errores en cada método o clase de forma individual antes de la integración de las mismas.

Las pruebas de caja negra, al igual que las de caja blanca, definen entradas y salidas esperadas, haciendo énfasis en los casos límite. A diferencia de las pruebas de caja blanca, no se tiene en cuenta la estructura interna del código.

Pruebas de integración

Las pruebas de integración tienen el objetivo de probar varios componentes de la aplicación una vez estos han sido integrados entre sí. Para esta aplicación, se ha decidido realizar pruebas de integración ascendentes. De este modo, cada componente se prueba tras integrarse con el inmediatamente inferior a él (de cuyos métodos hace uso directo). Se han realizado pruebas de integración para todos los componentes de la aplicación, una vez finalizadas las pruebas unitarias y de integración de los elementos inferiores.

Pruebas de regresión

Todas las pruebas previamente definidas (exceptuando la lectura de código) se han automatizado mediante código Java con el objetivo de que sirvan como pruebas de regresión. De este modo, este código puede volverse a ejecutar tras una modificación o ampliación del sistema para comprobar si se han introducido nuevos fallos en el mismo.

Pruebas de la interfaz de usuario

Para la interfaz de la aplicación, implementada como una página web, se han realizado pruebas de tipo web. La interfaz de usuario se ha verificado mediante la realización de pruebas de contenido. Estas pruebas tienen como objetivo determinar si la interfaz cuenta con todos los elementos necesarios para su correcto funcionamiento. Estos elementos son datos que la aplicación muestra al usuario o entradas de datos que el usuario debe introducir. Asimismo, se comprueba que aquellos elementos dinámicos de la interfaz cuya apariencia debe modificarse tras una determinada acción del usuario tienen el comportamiento esperado.

9.1.2. Desarrollo de las pruebas

A continuación se describe cuándo y cómo se han realizado estas pruebas y qué resultados se han obtenido de las mismas. Se detalla el procedimiento seguido para cada tipo de prueba realizada.

Inspección del código

La inspección del código se ha llevado a cabo inmediatamente después de la implementación de cada una de las clases que implementan los algoritmos de particionamiento y facturación. Las

pruebas han consistido en una lectura coherente del código que contienen por la alumna que lo ha programado. Estas pruebas han permitido detectar un número importante de errores de los tipos que se especifican a continuación:

- Errores sintácticos en las asignaciones o llamadas a funciones. Se habían asignado las variables equivocadas o se habían pasado parámetros incorrectos a las funciones. En estos casos una variable había sido usada en lugar de otra existente en el código.
- Errores de la implementación de los límites de particionamiento en el uso del álgebra modular. En este caso se habían implementado de forma incorrecta las operaciones de conjuntos definidas para el algoritmo de particionamiento.
- Errores en la asignación de descripciones a las unidades de facturación. Algunas unidades de facturación tenían nombres o informaciones incorrectas o correspondientes a otras unidades.
- Errores en la comprobación de pertenencia de los datos de uso a conjuntos de condiciones. En estos casos la comprobación se había realizado de forma incorrecta por errores sintácticos.

Pruebas unitarias de caja blanca

Para la realización de las pruebas de caja blanca, se ha seguido el procedimiento especificado en [26]. Este procedimiento consiste en la obtención del número de caminos independientes de un código estructurado como grafo. Para ello, se determina la complejidad ciclomática del mismo como $Aristas - Nodos + 2$. A continuación, se identifican dichos caminos independientes y se diseñan una o varias pruebas para cada uno de ellos, haciendo énfasis en los valores límite. Estas pruebas definen una entrada y una salida esperada para cada uno de ellos. La entrada se define de modo que trace el camino deseado, satisfaciendo o no cada bucle o condición. Estas pruebas han permitido detectar pequeños errores funcionales que producían una salida errónea. En el caso de los algoritmos de particionamiento y facturación, estos errores no habían sido detectados previamente mediante la inspección de código y se trataba, por tanto, de errores más sutiles.

Pruebas unitarias de caja negra

Para la realización de las pruebas de caja negra, se han definido casos de prueba válidos e inválidos para cada método a probar. Es decir, se han definido tanto casos en los que la salida esperada era exitosa como casos en los que debía ser errónea. Para cada uno de estos casos, se ha definido un plan de pruebas con la entrada y la salida esperada. Para el diseño de dicho plan de pruebas, se ha tratado de provocar en lanzamiento de todos los tipos de excepciones para cada

método. Asimismo, se ha tratado, para cada parámetro de entrada, de definir casos de prueba en los que tomase un valor límite.

Este plan de pruebas se ha traducido en clases de prueba JUnit. Se ha definido una clase de prueba para cada clase a probar y uno o varios métodos de prueba para cada método a probar. En JUnit se han definido automáticamente tanto la entrada como la salida esperada de cada caso (incluyendo los errores). Se han ejecutado automáticamente las pruebas, detectándose fallos o errores mediante la visualización de un informe de resultados. Estos errores o fallos se han localizado dentro del método correspondiente para su subsanación. Las pruebas de caja negra se han dado por finalizadas al ejecutarse correctamente todos los casos de prueba en JUnit.

Pruebas de integración

Las pruebas de integración han seguido una estrategia ascendente. Una vez finalizadas las pruebas unitarias, cada componente ha sido probado sobre todos aquellos de los que hacía uso. Para ello, se ha definido un plan de pruebas representando escenarios de uso de la aplicación restringidos al componente a probar.

Pruebas de la interfaz de usuario

La verificación de la interfaz de usuario se ha llevado a cabo mediante la prueba de escenarios de uso por parte de la desarrolladora. Estas pruebas han tenido como objetivo detectar errores en el correcto funcionamiento de la interfaz de usuario, integrada sobre el resto de componentes de la aplicación. Se han diseñado casos de prueba representantes del uso de esta aplicación por parte de los usuarios, intentando abarcar la mayor variedad de formas de uso y realizando pruebas sobre valores límite. Se ha comprobado que las acciones llevadas a cabo por el usuario en la interfaz realizaban los cambios esperados en los datos de la aplicación o en la salida ofrecida por la propia interfaz.

Dada la naturaleza dinámica de esta interfaz de usuario en el caso de algunos componentes de la misma, se ha verificado el correcto funcionamiento en este aspecto. El objetivo ha sido comprobar que cada acción llevada a cabo por el usuario ha desencadenado los cambios esperados en la interfaz.

9.2. Validación

El objetivo del proceso de validación es comprobar que se satisfacen los requisitos (tanto funcionales como no funcionales) definidos en el capítulo 5.

9.2.1. Estrategia de validación

Para la validación de esta aplicación, se ha comprobado cada uno de los requisitos desde el punto de vista de un usuario final. Esta tarea ha sido realizada primero por la desarrolladora y posteriormente en colaboración con individuos que no han participado en el proceso de desarrollo.

Dado que el usuario objetivo de este sistema cuenta con conocimientos de informática y telecomunicaciones, los individuos que han colaborado en estas pruebas pertenecen a dichas áreas. Estos usuarios, además, han probado la usabilidad del sistema mediante distintas métricas.

9.2.2. Desarrollo de la validación

El plan de validación ha consistido en la comprobación de cada requisito de la aplicación. Los requisitos que definen propiedades o características de la aplicación han sido comprobados mediante la realización de una acción que hiciese uso de dicha característica o propiedad. Los requisitos que definen restricciones sobre el sistema han sido comprobados mediante la realización de acciones que requerían de la ausencia de dicha restricción. Estos requisitos se han considerado como comprobados al no ser posible realizar ninguna de estas acciones.

La usabilidad de la interfaz de usuario ha sido comprobada con la colaboración de individuos del área informática. Se han definido dos métricas para la usabilidad.

- Tasa de acciones que no han sido capaces de realizar o han realizado incorrectamente en el primer uso de la aplicación. Esta métrica pretende medir la intuitividad en el uso de la aplicación, es decir, en qué medida puede ser usada sin conocimientos previos.
- Tasa de acciones que no han sido capaces de realizar o han realizado incorrectamente tras una explicación del funcionamiento de la aplicación y con la ayuda de una guía de usuario. Esta métrica pretende medir la facilidad de uso de la aplicación de un usuario entrenado para ello.

Tras las pruebas, se han obtenido para la segunda tasa valores más altos que para la primera. Los valores de la segunda tasa indican que la aplicación es usable para usuarios entrenados.

10

Evaluación

10.1. Evaluación de los usuarios

Distintos usuarios han evaluado el sistema tras su implementación y validación. Estos usuarios provienen del área informática y, por tanto, coinciden con el usuario objetivo del mismo.

Los usuarios han destacado que el tarificador cumple con los requisitos para la tarificación de Cloud Services. Además, todos ellos han notado que permite crear una gran variedad de modelos de precio para los mismos. Estos modelos de precio son flexibles y se adaptan a todas las necesidades que los usuarios han presentado.

Se ha valorado la posibilidad de crear múltiples planes de precio asociados a un mismo producto. Estos planes de precio se han definido como extensibles, dada la amplia selección de componentes de precio que se pueden combinar. Ha destacado la contratación selectiva, que permite a los usuarios elegir un plan de precio y determinadas características según sus preferencias.

Se ha considerado útil la facturación manual dado que permite conocer el total tarificado sobre los datos de uso en todo momento. Además, la tarificación automática permite al proveedor de servicios interactuar con el sistema únicamente de forma esporádica para la definición de productos y sus modelos de precio.

La interfaz del proveedor de servicios se ha considerado usable tras un aprendizaje inicial de periodo corto. Se ha propuesto incrementar la usabilidad de la interfaz mediante de herramientas

de edición y deshecho, así como la creación de una interfaz simplificada para usuarios menos avanzados. Asimismo, se ha destacado la facilidad de creación de un modelo de precio sencillo a pesar de la posibilidad de definir modelos de precio altamente complejos.

La interfaz del cliente de servicios, por el contrario, se ha considerado completamente sencilla y usable dados los pocos pasos que son necesarios para contratar un producto y la intuitividad del proceso. Toda la información de los modelos de precio es sintetizada para que el usuario la visualice y seleccione.

10.2. Beneficios de la aplicación

El modelo que se ha definido en este Trabajo de Fin de Grado presenta un lenguaje genérico de definición de productos y modelos de precio y un algoritmo de tarificación integrables en cualquier sistema para la tarificación.

La aplicación que se ha construido sobre dicho modelo ha permitido definir, ofertar y tarificar Cloud Services implementados sobre distintas capacidades ofrecidas por Telefónica como M2M, envío de SMS y MMS, o de control de gasto. Este sistema de tarificación permite un uso internacional dada la gestión de divisas y países que realiza en los planes de precio.

Este sistema permite además la inserción de datos de uso provenientes de distintos contextos y gestionados por distintas aplicaciones. Permite definir distintas condiciones según las que tarificar estos datos de uso, tanto temporales como de uso acumulado. Esto permite representar la mayoría de los modelos de precio vigentes actualmente en la tarificación de Cloud Services.

El modelo definido es integrable con distintos sistemas de pago y controladores de límite de gasto. El tarificador que se ha implementado se integrará con un sistema real de cobro de Telefónica y con un servicio de límite de gasto del área financiera de Telefónica I+D.

10.3. Comparación con otras herramientas

En la sección 3 de este documento se han descrito algunas aplicaciones de tarificación existentes en el mercado así como algunos modelos teóricos. A continuación se compara el tarificador implementado en este Trabajo de Fin de Grado con los anteriormente descritos.

Este tarificador describe un modelo genérico similar a la especificación USDL. En cambio, el modelo aquí propuesto describe un lenguaje de definición de precios más cercano al lenguaje humano, de forma que un usuario con conocimientos de informática puede utilizarlo con facilidad. Esto, en cambio, no limita los precios que este lenguaje permite definir.

A diferencia de los sistemas descritos en la sección 3, este sistema de tarificación está integrado dentro de un Marketplace de Cloud Services. Sin embargo, el modelo sobre el que está basado es adaptable a cualquier otro contexto.

El sistema FreeSide integra la monitorización y obtención de datos de uso dentro del mismo. En cambio, este sistema permite obtener los datos de uso de múltiples orígenes, de forma que este proceso lo gestione la propia aplicación que se usa.

Al igual que los otros sistemas estudiados, este sistema no es de uso libre. El modelo en él presentado sí podría serlo, de forma que pudiese implementarse en un contexto independiente.

En lugar de usar un software de pagos con tarjeta de crédito como el sistema JBilling, este sistema es adaptable a cualquier sistema de pagos, al que sólo debe proporcionarle el cliente y la cuantía a facturar. Como prueba de ello, este sistema se integrará con un sistema de pagos de Telefónica en el futuro.

11

Conclusiones

Este Trabajo de Fin de Grado ha consistido en el desarrollo completo de un proceso de Ingeniería del Software para la obtención de una aplicación informática. Se trata de un sistema de tarificación electrónica de Cloud Services integrado en el Marketplace del proyecto de investigación TESLA. Dicho proyecto consiste en la implementación de una plataforma global de distribución de servicios de telecomunicaciones basados en la red de Telefónica. El proceso realizado por la estudiante ha consistido en el análisis del problema e investigación del estado del arte, definición de la aplicación a construir, diseño de la solución, implementación de la misma, verificación y validación.

Para ello, se ha partido de una identificación de las necesidades de usuario y de mercado y de las restricciones impuestas sobre este sistema para su posterior integración. Se han analizado las necesidades y se ha definido un sistema de tarificación de servicios según el uso que cada cliente ha realizado de los mismos.

Este sistema permite a los proveedores de servicios definir sus servicios y modelos de precio asociados a los mismos que servirán para permitir su contratación y realizar una posterior facturación. El modelo de precio se compone de distintos planes que podrán ser contratados. Cada uno de estos planes define una duración del contrato, una frecuencia de facturación, una o varias monedas de cobro y distintos precios de facturación. Además, cada producto puede tener definidas distintas calidades de servicio que el cliente puede seleccionar durante la contratación.

Cada calidad puede tener asociado un descuento por incumplimiento del acuerdo de servicio.

A su vez, los clientes pueden contratar un servicio eligiendo un plan de precio y diferentes características. También pueden obtener posteriormente facturas del servicio según el uso que han realizado del mismo. Estas facturas se generarán automáticamente con la frecuencia definida en el plan de precios. Además, será posible obtener un cálculo intermedio del total acumulado durante un periodo. Cada contrato puede cancelarse antes de su finalización y en este caso, suponer una penalización económica para el cliente dependiente del momento de cancelación.

El sistema cuenta además con una interfaz de usuario que permite hacer uso de la funcionalidad descrita de manera interactiva. Esta interfaz se ha implementado de forma que sea usable para un usuario con conocimientos informáticos.

La investigación del estado del arte ha detectado la escasez de sistemas de tarificación genérica en el mercado actual. La mayoría de los sistemas existentes son específicos para la tarificación en una área concreta. El único sistema genérico de tarificación de servicios encontrado es de pago y de uso externo. Por tanto, no puede ser integrado dentro de otra plataforma.

Este Trabajo de Fin de Grado proporciona un diseño genérico para la tarificación electrónica. Este diseño proporciona múltiples tipos de modelos de precio comunes en el mercado actual pero de los que no existe una implementación. La implementación de este diseño que se ha desarrollado para el proyecto TESLA proporciona una forma sencilla y a la vez genérica de tarificar los servicios ofrecidos por Telefónica en dicho proyecto. Se permiten múltiples modelos de precio para una tarificación precisa de los mismos.

Durante el desarrollo de este Trabajo de Fin de Grado, se han podido poner en práctica las técnicas y métodos de Ingeniería del Software aprendidos en la carrera mediante la realización de un proceso de desarrollo de software completo. Además, se han adquirido conocimientos sobre tecnologías no estudiadas previamente que han sido usadas para el desarrollo de este trabajo.

Se han diseñado modelos de datos basados en XML, que se han definido mediante el lenguaje XSD. Estos modelos se han gestionado desde código Java mediante la herramienta JAXB. Se ha utilizado también el formato JSON para el almacenamiento, realizando lecturas del mismo desde Java. Además, se han utilizado las herramientas Hibernate y Spring para la conexión con bases de datos y la gestión del modelo vista-controlador respectivamente. Aunque estas herramientas no habían sido utilizadas previamente en la universidad, su uso está muy extendido entre los desarrolladores informáticos actualmente.

Todas las tareas y funcionalidades aquí comentadas han sido íntegramente definidas e implementadas por la estudiante autora de este trabajo desde el inicio al producto final.

Por todo esto, este trabajo, además de resultar de utilidad para un proyecto de investigación, ha permitido a la estudiante aprender nuevas técnicas de diseño y desarrollo informático así como poner en práctica las aprendidas durante los estudios.

12

Líneas de trabajo futuras

Durante la realización de este Trabajo de Fin de Grado se ha implementado el sistema de tarificación definido y se ha integrado dentro del Marketplace del proyecto TESLA. Además del trabajo realizado, se considera que hay otras tareas o áreas de trabajo que serían de interés en el futuro con el objetivo de ampliar este sistema.

Ampliación de la interfaz de usuario

La interfaz de usuario que se ha implementado permite realizar las tareas básicas implementadas en el tarificador como creación de productos, contratación u obtención de factura. Sin embargo, otras funcionalidades más específicas como la edición de productos o modelos de precio que sí han sido implementadas, no han sido incluidas en la interfaz de usuario de este prototipo. La creación de estas interfaces permitiría ampliar el sistema y dotarlo de una mayor funcionalidad y usabilidad para los usuarios.

Exportación de una API genérica de tarificación

Aunque el diseño aquí definido es genérico y válido para la tarificación en cualquier contexto, su implementación se ha centrado en la tarificación específica de servicios para el Marketplace del proyecto TESLA. En un futuro, la exportación de la funcionalidad de tarificación a una

API genérica permitiría tarificar en otros contextos distintos a este y obtener una factura para cualquier conjunto de productos, modelos de precio, contratos y datos de uso definidos en el formato presentado para este sistema de tarificación.

Conexión con un sistema real de cobro

Este sistema de tarificación obtiene facturas para los distintos contratos generados en el Marketplace del proyecto TESLA. Sin embargo, estas facturas no constituyen un cobro real. La conexión de este sistema con la API de pagos existente en Telefónica permitirá realizar un cobro efectivo de las facturas proporcionadas por el tarificador. Esto permitirá utilizar este sistema de tarificación para el cobro de cualquier servicio de internet definido en el lenguaje presentado para el sistema.

Integración con una API de control de gasto

Este sistema de tarificación proporciona a los usuarios la opción de establecer un límite de gasto para sus contratos. Asimismo, los proveedores de servicios pueden acceder a esta información para hacer uso de ella. La integración de este tarificador con la API de control de gasto existente en Telefónica I+D permitirá hacer efectiva la limitación impuesta por el usuario de manera sencilla.

Gestión de impuestos

Los modelos de precio definidos en este tarificador se componen de precios básicos sin tomar en consideración el cobro de impuestos. Dado que todas las transacciones comerciales reales conllevan la aplicación de impuestos, sería interesante extender el modelo de precio definido en este Trabajo de Fin de Grado para que permitiese gestionar distintos tipos de impuestos. Esto facilitaría su uso en transacciones comerciales reales.

Bibliografía

[1] Reed K. Holden Thomas T. Nagle. *Estrategia y Tácticas de Precios*. Prentice Hall, 2002.

[2] Kent B. Monroe. *Política de precios*. McGraw-Hill, 1992.

[4] Alistair Barros et al. "Unified Service Description Language 3.0 (USDL) Overview". En: (2011).

[5] Alistair Barros et al. "Unified Service Description Language (USDL) Pricing Module". En: (2011).

[6] Igor Ruiz-Agundez et al. "Tarificación flexible de servicios en Internet". En: (2010).

[7] Michael Hanrahan et al. "Bulk Data Collection". En: (2011).

[8] Emiliano Conde et al. *JBilling Telecom Guide*. 2010.

[22] 3GPP. "ETSI TS 132 297, Charging Data Record (CDR) file format and transfer". En: (2012).

[24] B. Shamkant Navathe Elmasri Ramez. *Fundamentals of Database Systems*. Addison-Wesley, 2000.

[26] Bolaños et al. *Pruebas de Software y JUnit*. Prentice-Hall, 2008.

Referencias

[3] *Dropbox*. URL: http://www.dropbox.com/billing/.

[9] *CDR (Charging Data Record)*. URL: http://www.3gpp.org/DynaReport/32297.htm.

[10] *FreeSide*. URL: http://www.freeside.biz/freeside/.

[11] *MySQL*. URL: http://www.mysql.com/.

[12] *Hibernate*. URL: http://www.hibernate.org/.

[13] *Hibernate Query Language (HQL)*. URL: http://docs.jboss.org/hibernate/orm/3.3/reference/en/html/queryhql.html.

[14] *Java Database Connectivity (JDBC)*. URL: http://www.oracle.com/technetwork/java/javase/jdbc/index.html.

[15] *Extensible Markup Language (XML)*. URL: http://www.w3.org/XML/.

[16] *XML Schema (XSD)*. URL: http://www.w3.org/XML/Schema.

[17] *Java Architecture for XML Binding (JAXB)*. URL: https://jaxb.java.net/.

[18] *Spring*. URL: http://spring.io/.

[19] *Java Server Pages Technology (JSP)*. URL: http://www.oracle.com/technetwork/java/javaee/jsp/index.html.

[20] *JasperReports*. URL: http://community.jaspersoft.com/project/jasperreports-library.

[21] *JSON specification*. URL: http://www.json.org/.

[23] *3GPP*. URL: http://www.3gpp.org/.

[25] *JUnit Framework*. URL: http://junit.org/.

Los enlaces que aquí se incluyen son manuales e información extendida acerca de herramientas mencionadas en el texto.

A

Definición de los XML

A continuación se incluyen en código XSD las definiciones de los XML de productos, modelos de precio y contratos.

XML de producto

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="producto">
  <xs:complexType>
    <xs:sequence>
    <xs:element name="componente" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="params" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="parametro" minOccurs="1" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:choice>
                      <xs:element name="tipo_discreto" minOccurs="1" maxOccurs="1">
                        <xs:complexType>
                          <xs:sequence>
```

```
                        <xs:element name="valor" type="xs:string" minOccurs="1" maxOccurs="
                            unbounded"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="tipo_int" minOccurs="1" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="min" type="xs:int"/>
                        <xs:element name="max" type="xs:int"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="tipo_doub" minOccurs="1" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="min" type="xs:double"/>
                        <xs:element name="max" type="xs:double"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:choice>
                <xs:attribute name="nombre" type="xs:string" use="required"/>
                <xs:attribute name="unidad" type="xs:string" use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="unidades" minOccurs="1" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="unidad" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="nombre" type="xs:string" use="required"/>
    <xs:attribute name="id" type="xs:int" use="required"/>
  </xs:complexType>
 </xs:element>
 </xs:sequence>
 <xs:attribute name="id" type="xs:int" use="required"/>
 <xs:attribute name="nombre" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

XML de modelo de precio

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="modeloprecio">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="plan" minOccurs="1" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="monedas" minOccurs="1" maxOccurs="1">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="moneda" minOccurs="1" maxOccurs="unbounded">
         <xs:complexType>
          <xs:sequence>
          <xs:element name="id" minOccurs="1" maxOccurs="1">
           <xs:simpleType>
            <xs:restriction base="xs:string">
             <xs:pattern value="[A-Z][A-Z][A-Z]"/>
            </xs:restriction>
           </xs:simpleType>
          </xs:element>
          <xs:element name="pais" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="precio_base" minOccurs="1" maxOccurs="1">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="inicial" type="xs:double" minOccurs="1" maxOccurs="1"/>
        <xs:element name="precio_periodo" type="precio-periodo" minOccurs="1" maxOccurs="1"/>
        <xs:element name="periodo" minOccurs="1" maxOccurs="1">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="cantidad" type="xs:int" minOccurs="1" maxOccurs="1"/>
           <xs:element name="tipo" type="tipo-periodo" minOccurs="1" maxOccurs="1"/>
          </xs:sequence>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="componente" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
```

```
<xs:sequence>
 <xs:element name="precio" minOccurs="1" maxOccurs="1">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="condicion" type="cond" minOccurs="1" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="params" minOccurs="0" maxOccurs="1">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="parametro" minOccurs="1" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
      <xs:element name="opcion" minOccurs="1" maxOccurs="unbounded">
       <xs:complexType>
        <xs:sequence>
         <xs:choice>
          <xs:element name="valor" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="rango_int" type="rango-int" minOccurs="1" maxOccurs="1"/>
          <xs:element name="rango_doub" type="rango-doub" minOccurs="1" maxOccurs="1"/>
         </xs:choice>
         <xs:element name="precio" minOccurs="1" maxOccurs="1">
          <xs:complexType>
           <xs:sequence>
            <xs:element name="fijo" type="xs:double" minOccurs="0" maxOccurs="1"/>
            <xs:element name="precio_periodo" type="precio-periodo" minOccurs="0" maxOccurs
                ="1"/>
            <xs:element name="condicion" type="cond" minOccurs="0" maxOccurs="unbounded"/
                >
           </xs:sequence>
          </xs:complexType>
         </xs:element>
         <xs:element name="incumplimiento" minOccurs="0" maxOccurs="1">
          <xs:complexType>
           <xs:sequence>
            <xs:element name="cond" minOccurs="1" maxOccurs="unbounded">
             <xs:complexType>
              <xs:sequence>
               <xs:sequence minOccurs="0" maxOccurs="1">
                <xs:element name="rango_doub" type="rango-doub" minOccurs="0" maxOccurs=
                    "1"/>
               </xs:sequence>
               <xs:element name="value" type="xs:double" minOccurs="1" maxOccurs="1"/>
               <xs:element name="unidad" type="xs:string" minOccurs="1" maxOccurs="1"/>
              </xs:sequence>
             </xs:complexType>
            </xs:element>
```

```
                    </xs:sequence>
                   </xs:complexType>
                  </xs:element>
                  </xs:sequence>
                 </xs:complexType>
                </xs:element>
               </xs:sequence>
               <xs:attribute name="nombre" type="xs:string" use="required"/>
              </xs:complexType>
             </xs:element>
            </xs:sequence>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="nombre" type="xs:string" use="required"/>
         <xs:attribute name="id" type="xs:int" use="required"/>
        </xs:complexType>
       </xs:element>
       <xs:element name="duracion" minOccurs="1" maxOccurs="1">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="cantidad" type="xs:int" minOccurs="1" maxOccurs="1"/>
          <xs:element name="cancelacion" minOccurs="0" maxOccurs="1">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="fijo" type="xs:double" minOccurs="0" maxOccurs="1"/>
             <xs:element name="precio_periodo" type="precio-periodo" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
           </xs:complexType>
          </xs:element>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
       <xs:element name="gasto_minimo" type="xs:double" minOccurs="0" maxOccurs="1"/>
       <xs:element name="gasto_maximo" type="xs:double" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:int" use="required"/>
      <xs:attribute name="nombre" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:int" use="required"/>
   </xs:complexType>
   </xs:element>
   <xs:simpleType name="tipo-periodo">
   <xs:restriction base="xs:string">
    <xs:enumeration value="dia"/>
    <xs:enumeration value="semana"/>
```

```
  <xs:enumeration value="mes"/>
  <xs:enumeration value="anno"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="dia-semana">
 <xs:restriction base="xs:string">
  <xs:enumeration value="lunes"/>
  <xs:enumeration value="martes"/>
  <xs:enumeration value="miercoles"/>
  <xs:enumeration value="jueves"/>
  <xs:enumeration value="viernes"/>
  <xs:enumeration value="sabado"/>
  <xs:enumeration value="domingo"/>
 </xs:restriction>
</xs:simpleType>
 <xs:complexType name="rango-int">
 <xs:sequence>
  <xs:element name="min" type="xs:int" minOccurs="1" maxOccurs="1"/>
  <xs:element name="max" type="xs:int" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
 </xs:complexType>
  <xs:complexType name="rango-doub">
 <xs:sequence>
  <xs:element name="min" type="xs:double" minOccurs="1" maxOccurs="1"/>
  <xs:element name="max" type="xs:double" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
 </xs:complexType>
<xs:complexType name="hora">
 <xs:sequence minOccurs="1" maxOccurs="1">
  <xs:element name="min" type="xs:time" minOccurs="1" maxOccurs="1"/>
  <xs:element name="max" type="xs:time" minOccurs="1" maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>
 <xs:complexType name="precio-periodo">
  <xs:sequence>
  <xs:element name="cond" minOccurs="1" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="rango" type="rango-int" minOccurs="0" maxOccurs="1"/>
     <xs:element name="value" type="xs:double"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 </xs:complexType>
 <xs:complexType name="cond">
 <xs:sequence>
  <xs:choice minOccurs="0" maxOccurs="1">
```

```
  <xs:element name="consumo_actual" type="rango-int" minOccurs="1" maxOccurs="1"/>
  <xs:element name="consumo_total" minOccurs="1" maxOccurs="1">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="total" type="rango-int" minOccurs="0" maxOccurs="1"/>
     <xs:element name="hora" type="hora" minOccurs="0" maxOccurs="1"/>
     <xs:element name="dia" minOccurs="0" maxOccurs="1">
      <xs:complexType>
       <xs:sequence minOccurs="1" maxOccurs="1">
        <xs:element name="min" type="dia-semana" minOccurs="1" maxOccurs="1"/>
        <xs:element name="max" type="dia-semana" minOccurs="1" maxOccurs="1"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:choice>
 <xs:element name="value" type="xs:double" minOccurs="1" maxOccurs="1"/>
 <xs:element name="unidad" type="xs:string" minOccurs="1" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

XML de contrato

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="contrato">
 <xs:complexType>
  <xs:sequence>
  <xs:element name="producto" type="xs:long" minOccurs="1" maxOccurs="1"/>
  <xs:element name="plan" type="xs:string" minOccurs="1" maxOccurs="1"/>
  <xs:element name="componente" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
   <xs:sequence>
    <xs:element name="parametro" minOccurs="1" maxOccurs="unbounded">
     <xs:complexType>
      <xs:choice>
       <xs:element name="valor_int" type="xs:int"/>
       <xs:element name="valor_doub" type="xs:double"/>
       <xs:element name="valor" type="xs:string"/>
      </xs:choice>
      <xs:attribute name="nombre" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
```

```
      </xs:sequence>
      <xs:attribute name="nombre" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="limite_gasto" type="xs:double" minOccurs="0" maxOccurs="1"/>
    <xs:element name="duracion" minOccurs="1" maxOccurs="1">
      <xs:complexType>
      <xs:sequence>
        <xs:element name="desde" type="xs:date" minOccurs="1" maxOccurs="1"/>
        <xs:element name="hasta" type="xs:date" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
      </xs:complexType>
    </xs:element>
    </xs:sequence>
  </xs:complexType>
  </xs:element>
</xs:schema>
```

B

Interfaz de usuario: Maquetas

En este anexo se muestran las maquetas de la interfaz de usuario que se implementará como parte de este Trabajo de Fin de Grado. Se trata de una interfaz web que se integrará dentro de la interfaz del Marketplace del proyecto TESLA.
Los componentes de la interfaz cuyas maquetas se muestran a continuación serán completamente desarrollados por la estudiante haciendo uso de las herramientas JSP y Javascript.

B.1. Definición de producto

La interfaz de definición de producto cuya maqueta se muestra en la figura B.1 permitirá a los usuarios proveedores de servicios definir dinámicamente un producto sin necesidad de editar un fichero XML. Para ello, podrán añadir o eliminar componentes, a los que a su vez podrán añadir o eliminar parámetros y unidades. La interfaz no permitirá crear productos sin al menos un componente, y al menos una unidad para cada componente. Por último, se traducirá la definición realizada por el usuario a un fichero XML que se almacenará en la base de datos.

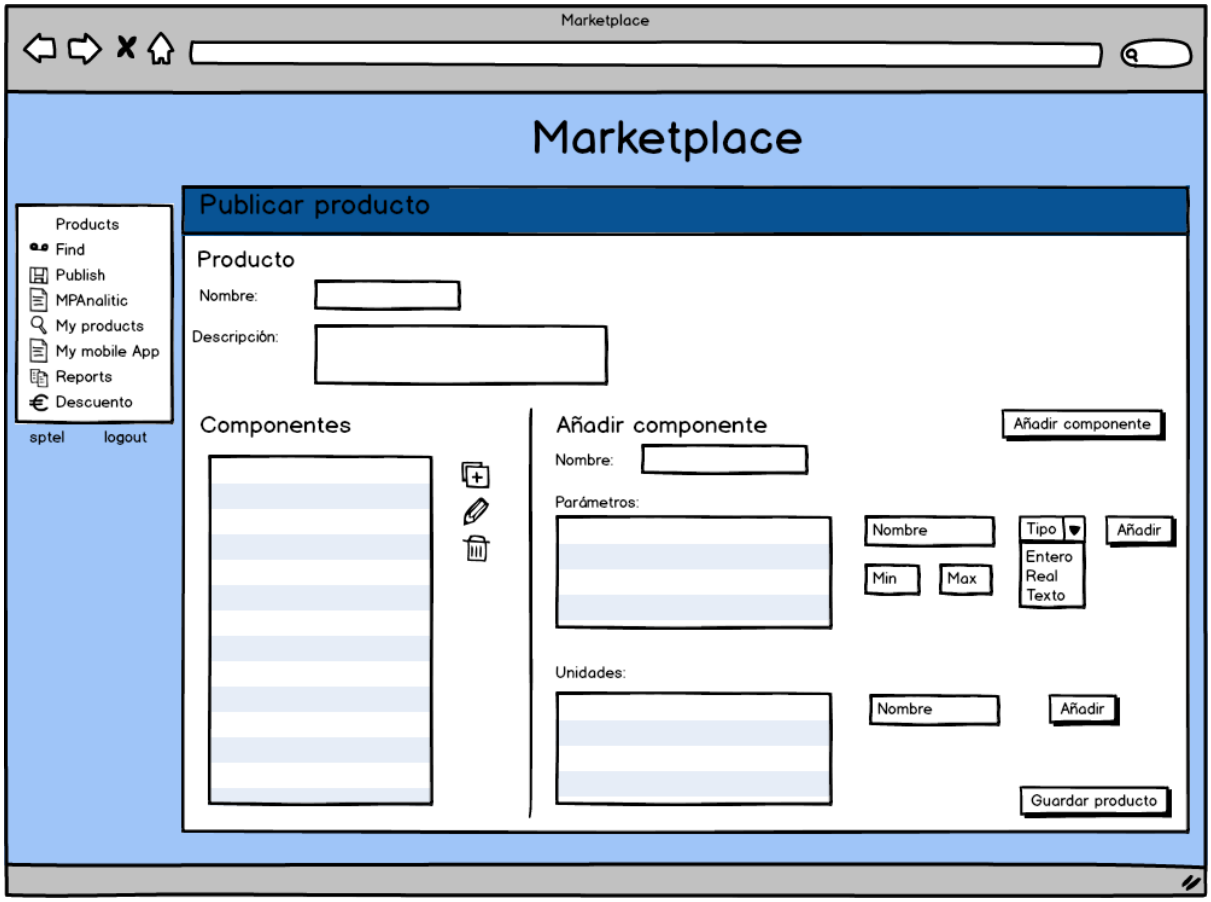

Figura B.1: Maqueta de la interfaz de definición de productos.

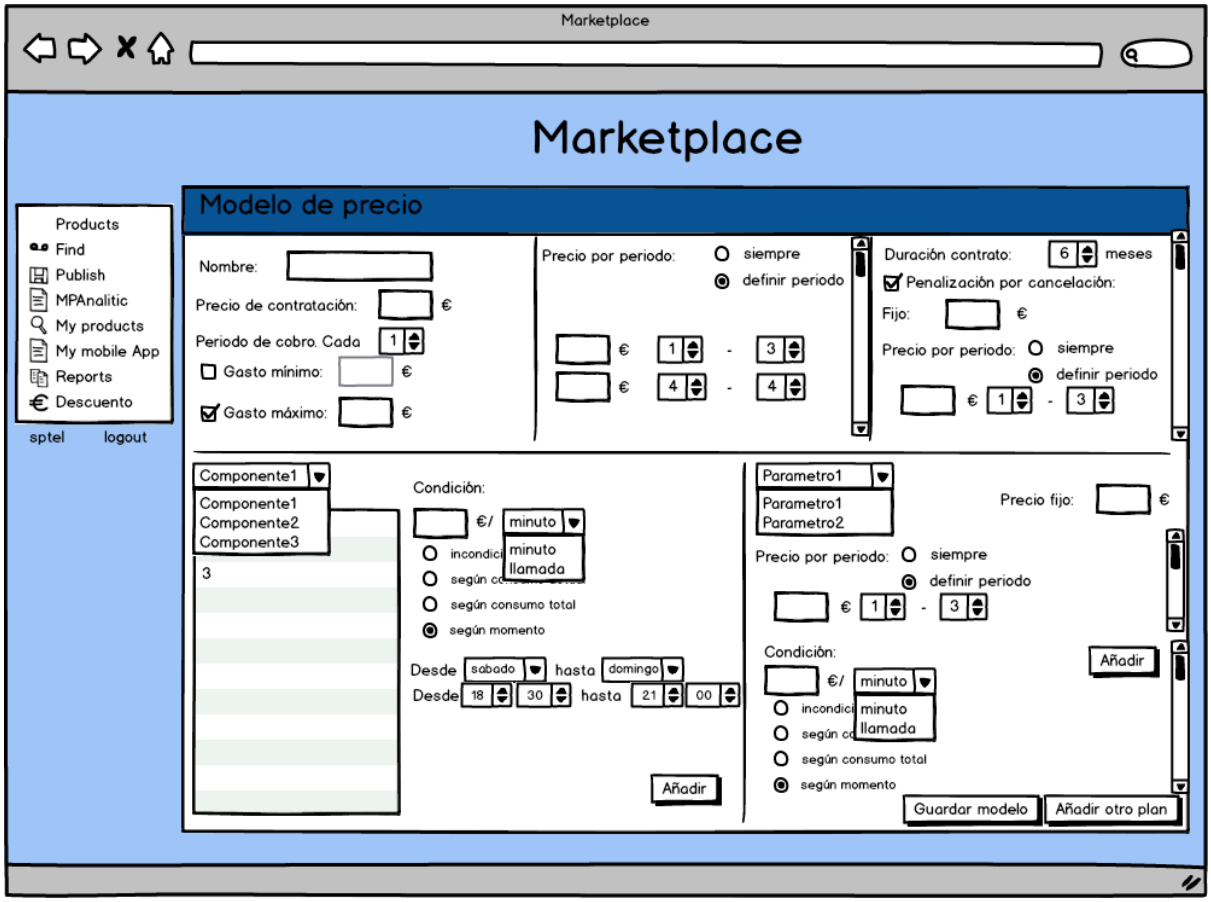

Figura B.2: Maqueta de la interfaz de definición de modelo de precio.

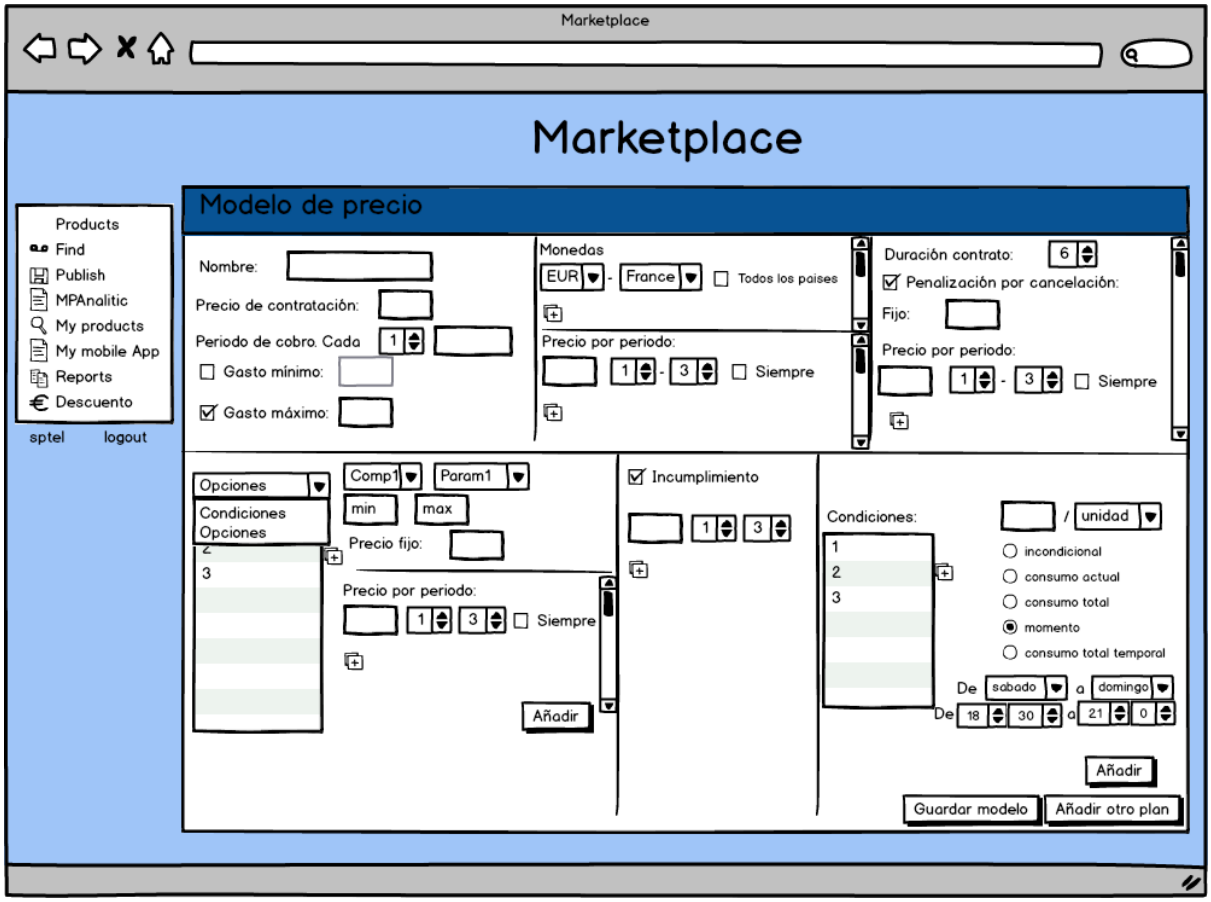

Figura B.3: Maqueta de la interfaz de definición de modelo de precio.

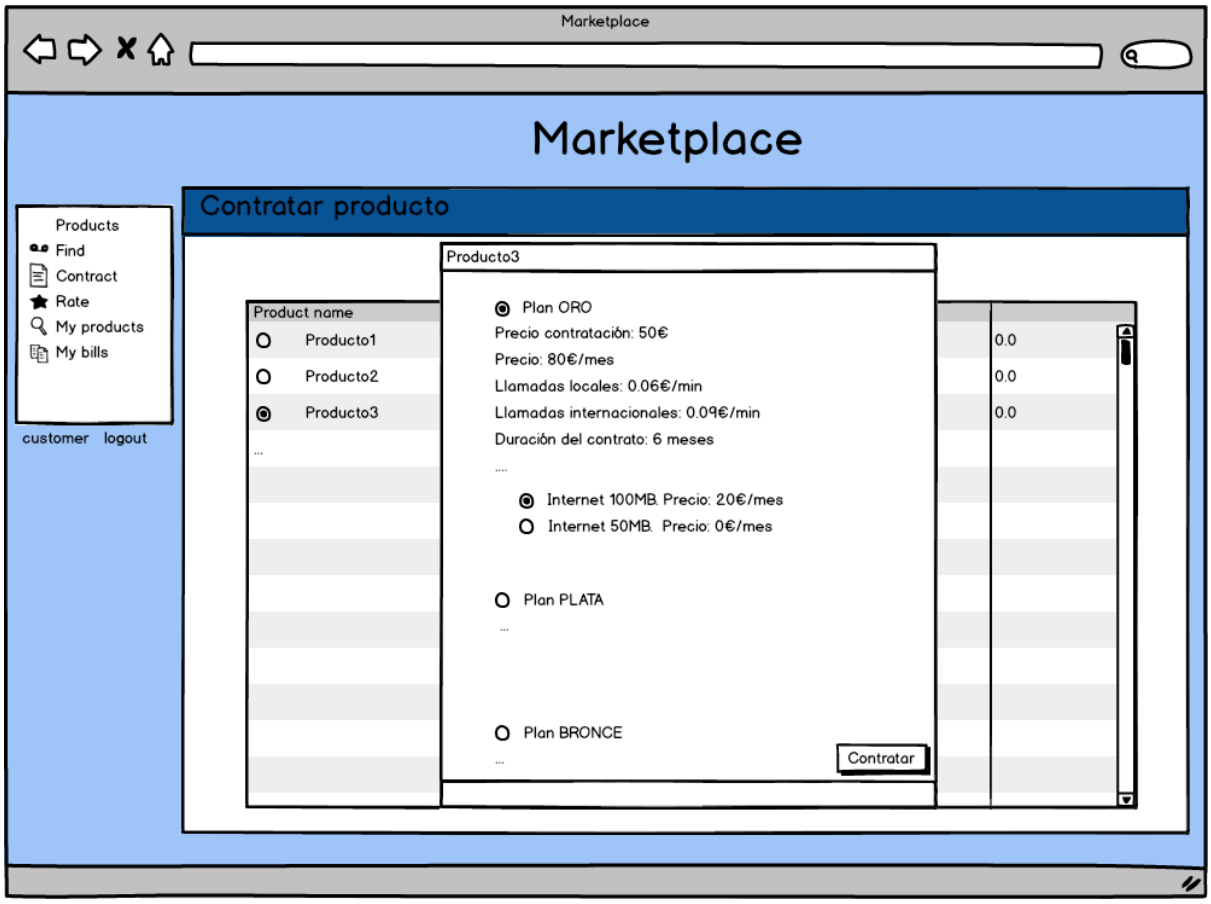

Figura B.4: Maqueta de la interfaz de contratación de producto.

B.2. Definición de modelo de precio

La interfaz de definición de modelo de precio cuya maqueta se muestra en las figuras B.2 y B.3 permitirá a los usuarios proveedores de servicios definir dinámicamente un modelo de precio sin necesidad de editar un fichero XML. La interfaz permite definir múltiples planes y, para cada uno de ellos, asignar el precio de contratación y precio por periodo del mismo. Asimismo, permite definir los tipos de divisa a los que aplica el plan, la duración del mismo y la penalización por cancelación. Para cada componente definido en el producto asociado, es posible añadir condiciones de precio según distintos criterios. Asimismo, para cada parámetro, es posible definir un modelo de precio para sus posibles valores, así como una penalización por incumplimiento. La interfaz implementada se adaptará a la estructura en forma de árbol del archivo XML. Se traducirá la definición realizada por el usuario a un fichero XML que se almacenará en la base de datos.

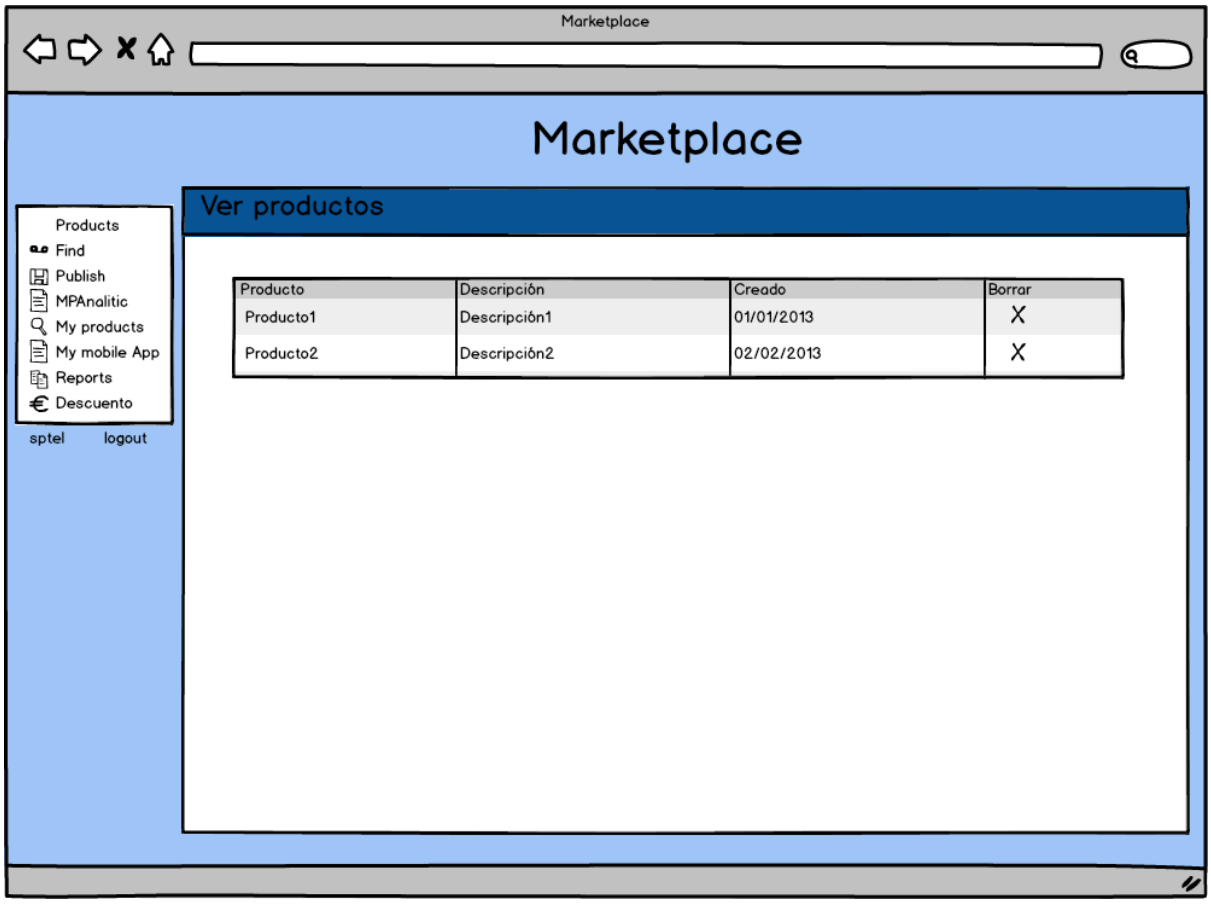

Figura B.5: Maqueta de la interfaz de visualización de productos.

B.3. Contratación

La interfaz de contratación de productos cuya maqueta se muestra en la figura B.4, visible desde la interfaz de cliente, permitirá contratar un producto de forma intuitiva. Esta contratación generará internamente un XML de contrato. Para ello, el usuario puede, desde la interfaz, elegir un plan de precios para el producto a contratar. Sólo le serán ofertados los planes de precios definidos para su país y su moneda de pago. Debe seleccionar también un valor para cada parámetro variable, y además, puede establecer un límite de gasto si lo desea.

B.4. Visualización y borrado de productos

En la figura B.5 se muestra la interfaz de visualización y borrado de productos para proveedores de servicios. Desde esta interfaz el proveedor de servicios puede ver todos los productos que ha definido y borrar alguno si lo desea. Este borrado supone la cancelación (sin penalización económica) de todos los contratos realizados sobre dicho producto.

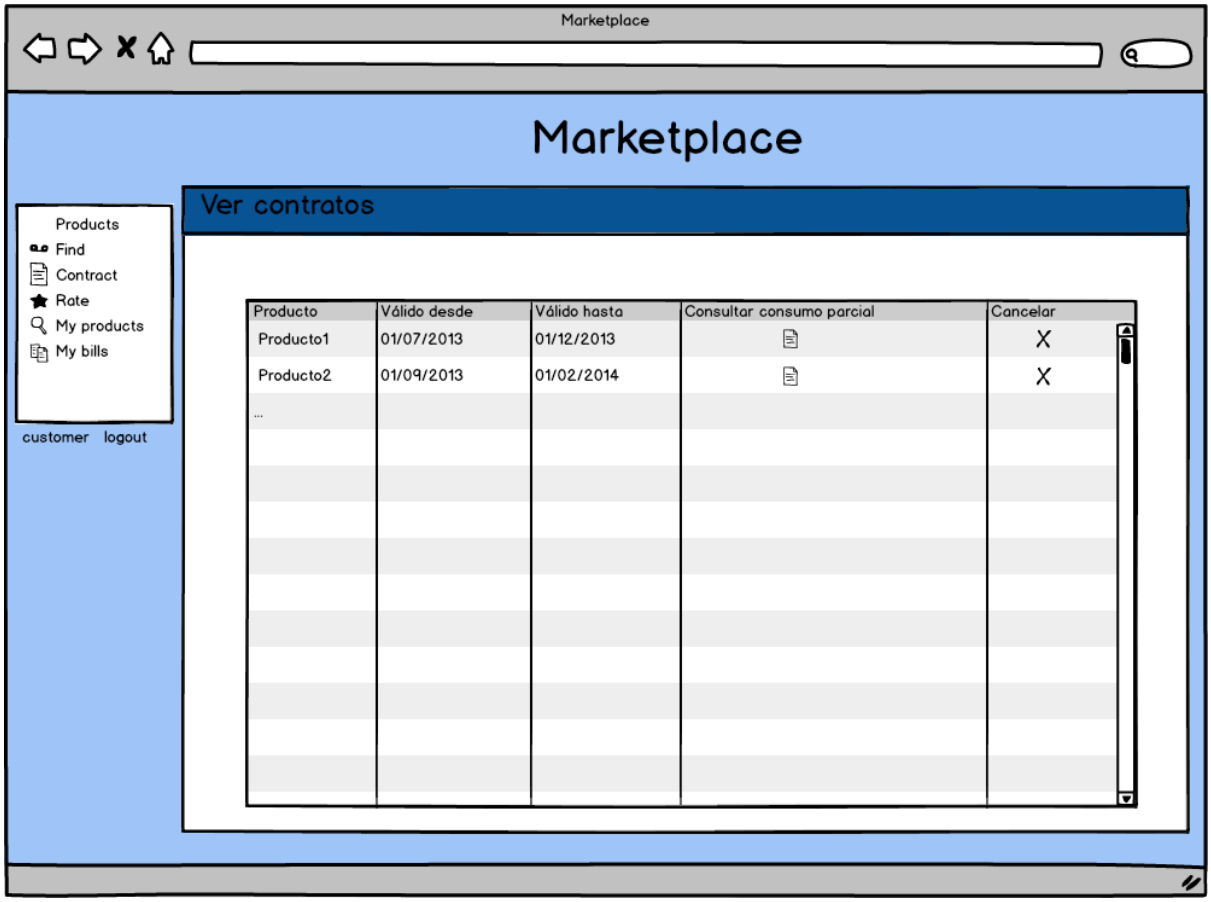

Figura B.6: Maqueta de la interfaz de visualización de contratos.

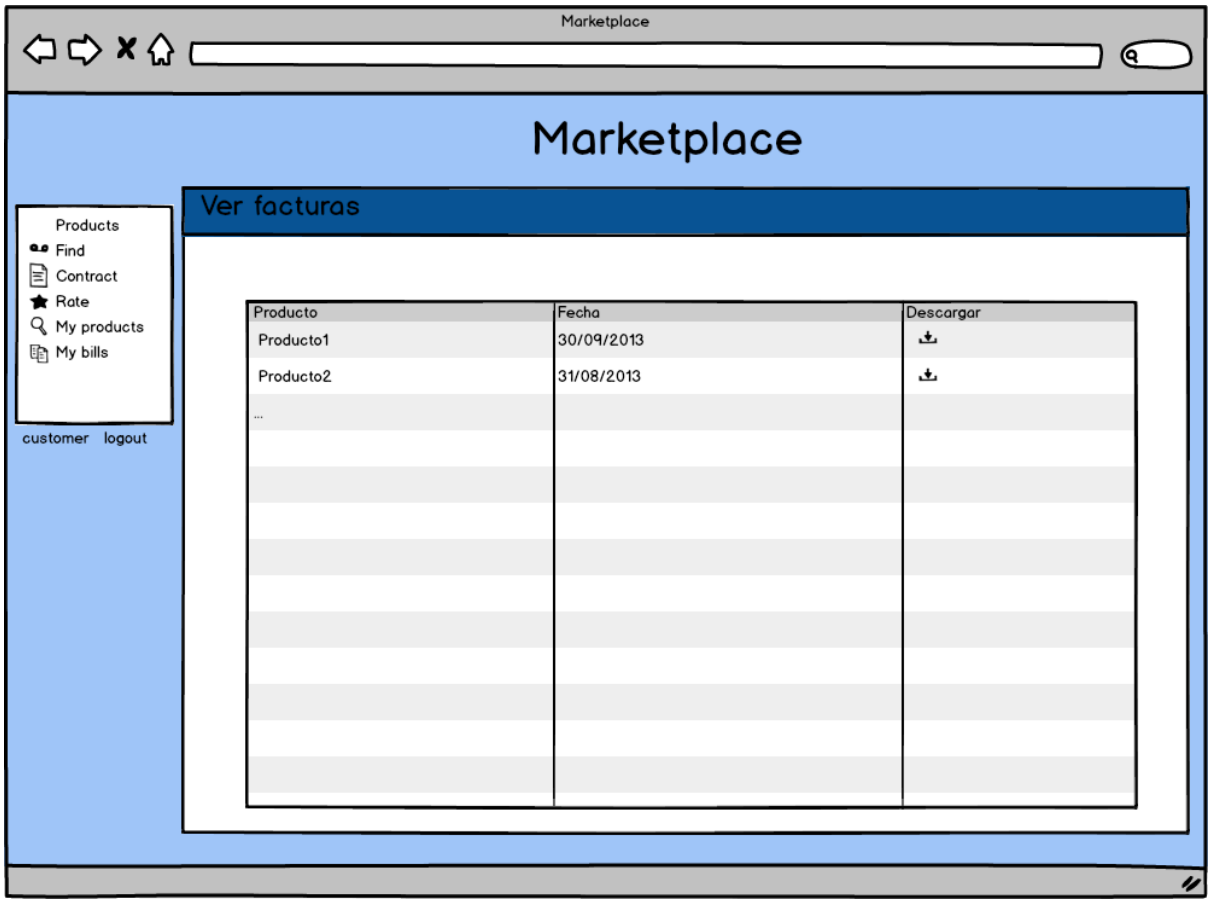

Figura B.7: Maqueta de la interfaz de visualización de facturas automáticas.

B.5. Visualización y cancelación de contratos

La interfaz de visualización de contratos, cuya maqueta se muestra en la figura B.6 permitirá a un cliente visualizar todos sus contratos activos, consultar el gasto parcial asociado a los mismos durante el periodo actual. Esta interfaz también permite al usuario cancelar cualquiera de sus contratos.

B.6. Visualización de facturas automáticas

En la figura B.7 se muestra la interfaz de visualización y descarga de facturas automáticas para clientes. Desde esta interfaz el cliente puede obtener sus facturas en formato PDF para todos sus contratos actuales o pasados generadas automáticamente al finalizar cada periodo.

C

Ejemplos de definición de productos, precios y contratos

En este anexo se muestran ejemplos de definición de productos, modelos de precio y contratos en el lenguaje de modelado definido sobre XML.

Ejemplo de definición de producto

A continuación se describe un ejemplo de producto definido en esta estructura.

Tomamos como ejemplo un servicio de control de gasto de llamadas que permite realizar dos funciones básicas: imponer un límite de gasto y consultar la cantidad restante hasta alcanzar el límite. Dicha consulta puede ser en tiempo real (informando del valor exacto en el momento de la consulta) o diferido (pudiendo informar de un valor desactualizado por hasta 10 minutos).

Además, la consulta puede realizarse de forma única (realizando una única petición de información en un momento dado) o de forma continua (visualizando información continua durante un periodo de tiempo). Por tanto, este producto cuenta con dos componentes: Imposición de límite y Consulta. El componente de consulta tiene un parámetro, tiempo real, que puede ser verdadero o falso, y puede medirse en dos unidades distintas: peticiones o segundos.

A continuación podemos ver el fichero XML de definición de este producto.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <producto id="1" nombre="Control de gasto">
   <componente nombre="Imposicion de limite" id="0">
       <unidades>
           <unidad>imposicion</unidad>
       </unidades>
   </componente>
   <componente nombre="Consulta cantidad restante" id="1">
       <params>
           <parametro nombre="Tiempo real" unidad="seleccionado">
               <tipo_discreto>
                   <valor>si</valor>
                   <valor>no</valor>
               </tipo_discreto>
           </parametro>
       </params>
       <unidades>
           <unidad>peticion</unidad>
           <unidad>segundo</unidad>
       </unidades>
   </componente>
</producto>
```

Ejemplo de definición de un modelo de precio

Se muestra ahora un ejemplo sencillo de modelo de precio con un único plan para el producto definido previamente. Dicho plan se hace únicamente disponible para aquellos usuarios de España o Francia que realicen sus pagos en euros.

La contratación de este plan conlleva un pago inicial de 20€ y un precio mensual de 10€. Se realiza una facturación mensual durante un total de 12 meses.

Cada imposición de límite tiene un precio de 0,50€ a diario y 0,30€ los fines de semana. Las consultas tienen un precio de 0,10€ por petición si son puntuales o 0,01€ por segundo si son continuas. Si se selecciona información en tiempo real estos precios se duplican. Si se produjese un incumplimiento de dicha calidad en tiempo real, el usuario recibiría un descuento de 0,40€ por segundo o 0,04€ por petición.

Si el usuario cancela el contrato antes de la finalización del mismo, debe abonar un precio fijo de 60€, 5€ por cada mes restante entre los 6 últimos y 10€ por cada mes restante entre los 6 primeros. De esta forma, si quedan 7 meses de contrato, el usuario deberá abonar (60+5*6+10*1)€= 100 €

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<modeloprecio id="1">
  <plan id="0" nombre="Unico">
    <monedas>
      <moneda>
        <id>EUR</id>
        <pais>Spain</pais>
        <pais>France</pais>
      </moneda>
    </monedas>
      <precio_base>
        <inicial>20</inicial>
        <precio_periodo>
          <cond>
            <value>10</value>
          </cond>
        </precio_periodo>
        <periodo>
          <cantidad>1</cantidad>
          <tipo>mes</tipo>
        </periodo>
      </precio_base>
      <componente nombre="Imposicion de limite" id="0">
        <precio>
          <condicion>
            <value>0.50</value>
            <unidad>imposicion</unidad>
          </condicion>
          <condicion>
            <consumo_total>
              <dia>
                <min>sabado</min>
                <max>domingo</max>
              </dia>
            </consumo_total>
            <value>0.30</value>
            <unidad>imposicion</unidad>
          </condicion>
        </precio>
      </componente>
      <componente nombre="Consulta cantidad restante" id="1">
        <precio>
          <condicion>
            <value>0.10</value>
            <unidad>peticion</unidad>
          </condicion>
          <condicion>
            <value>0.01</value>
```

```
            <unidad>segundo</unidad>
        </condicion>
    </precio>
    <params>
        <parametro nombre="Tiempo real">
            <opcion>
                <valor>si</valor>
                <precio>
                    <fijo>10</fijo>
                    <condicion>
                        <value>0.20</value>
                        <unidad>peticion</unidad>
                    </condicion>
                    <condicion>
                        <value>0.02</value>
                        <unidad>segundo</unidad>
                    </condicion>
                </precio>
                <incumplimiento>
                    <cond>
                        <value>0.40</value>
                        <unidad>segundo</unidad>
                    </cond>
                    <cond>
                        <value>0.04</value>
                        <unidad>peticion</unidad>
                    </cond>
                </incumplimiento>
            </opcion>
            <opcion>
                <valor>no</valor>
                <precio>
                    <fijo>0</fijo>
                </precio>
            </opcion>
        </parametro>
    </params>
</componente>
<duracion>
    <cantidad>12</cantidad>
    <cancelacion>
        <fijo>60</fijo>
        <precio_periodo>
            <cond>
                <rango>
                    <min>1</min>
                    <max>6</max>
                </rango>
```

```
                <value>5</value>
            </cond>
            <cond>
                <rango>
                    <min>1</min>
                    <max>6</max>
                </rango>
                <value>10</value>
            </cond>
        </precio_periodo>
    </cancelacion>
</duracion>
</plan>
</modeloprecio>
```

Ejemplo de contrato

A continuación se muestra un ejemplo de XML de contrato para el producto de Control de Gasto definido previamente con el modelo de precio que se ha usado como ejemplo. Este contrato muestra que se ha elegido el plan *Único* para este producto. Dentro de este plan, se ha elegido la opción *si* para el parámetro *Tiempo real*. No aparece en el contrato el otro componente del que consta el producto dado que no tiene parámetros. El contrato muestra además la duración del mismo, que es de un año desde la adquisición del producto.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<contrato>
    <producto>1</producto>
    <plan>Unico</plan>
    <componente nombre="Consulta cantidad restante">
        <parametro nombre="Tiempo real">
            <valor>si</valor>
        </parametro>
    </componente>
    <duracion>
        <desde>2013-11-19</desde>
        <hasta>2014-11-19</hasta>
    </duracion>
</contrato>
```

D

Ejemplo de facturación periódica

En este anexo se muestran ejemplos de contrato y de modelo de precio de un servicio de envío de SMS. El producto contratado cuenta con un único componente, encargado de realizar dicho envío. Este componente se puede configurar para obtener un acuse de recibo tras cada envío, por un coste adicional.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<modeloprecio id="20">
  <plan id="1" nombre="Unico">
    <monedas>
      <moneda>
        <id>EUR</id>
      </moneda>
    </monedas>
    <precio_base>
      <inicial>99.90</inicial>
      <precio_periodo>
        <cond>
          <rango>
            <min>1</min>
            <max>4</max>
          </rango>
          <value>29.90</value>
        </cond>
        <cond>
          <rango>
```

```
            <min>5</min>
            <max>20</max>
          </rango>
          <value>49.90</value>
        </cond>
      </precio_periodo>
      <periodo>
        <cantidad>1</cantidad>
        <tipo>semana</tipo>
      </periodo>
    </precio_base>
    <componente nombre="SMS" id="0">
      <precio>
        <condicion>
          <value>1</value>
          <unidad>envio</unidad>
        </condicion>
      </precio>
      <params>
        <parametro nombre="Acuse de recibo">
          <opcion>
            <valor>si</valor>
            <precio>
              <fijo>30</fijo>
                <precio_periodo>
                    <cond>
                        <rango>
                            <min>1</min>
                            <max>3</max>
                        </rango>
                        <value>0</value>
                    </cond>
                    <cond>
                        <rango>
                            <min>4</min>
                            <max>20</max>
                        </rango>
                        <value>5</value>
                    </cond>
                </precio_periodo>
            </precio>
          </opcion>
          <opcion>
            <valor>no</valor>
            <precio>
              <fijo>0</fijo>
            </precio>
          </opcion>
```

```
        </parametro>
      </params>
    </componente>
    <duracion>
      <cantidad>20</cantidad>
    </duracion>
  </plan>
</modeloprecio>
```

```
<contrato>
    <producto>20</producto>
    <plan>Unico</plan>
    <componente nombre="SMS">
        <parametro nombre="Acuse de recibo">
            <valor>si</valor>
        </parametro>
    </componente>
    <duracion>
        <desde>2013-11-05</desde>
        <hasta>2014-02-18</hasta>
    </duracion>
</contrato>
```

La facturación periódica de este contrato durante la tercera semana tras la contratación del mismo, producirá dos líneas de facturación.

La primera línea corresponderá al cobro del plan elegido. Al tratarse de la tercera semana, el cobro será el definido para las semanas de 1 a 4: 29,90€.

La segunda línea corresponderá al cobro por la selección del valor sí para el Acuse de Recibo. Por tratarse de una semana entre la primera y la tercera, este cobro será de 0€.

E

Definición del formato SDR

A continuación se muestran los distintos tipos de formato SDR que se usan en este tarificador.

Ficheros SDR de eventos

Los ficheros SDR de eventos indican un suceso en una aplicación en un determinado instante del tiempo. Cuentan con los siguientes campos:

ContractId Identificador del contrato al que pertenecen los datos de uso.

Timestamp Marca temporal del instante en el que se ha producido el evento.

RecordType Tipo de SDR, que en este caso es NamedEvent.

EventName Nombre del evento que se ha producido.

Info Información adicional acerca del evento.

En este tarificador, el único evento definido es la cancelación de un contrato. A continuación se muestra un ejemplo de un fichero SDR de evento de cancelación.

```
{
    "ContractId":"550e8400-e29b-41d4-a716-446655440000",
    "Timestamp":"2013-11-03T18:00:00Z",
    "RecordType":"NamedEvent",
    "EventName":"Cancelacion",
    "Info":"AdditionalInfo"
}
```

Ficheros SDR de cantidades

Los ficheros SDR de cantidades indican el uso de una cantidad de un elemento de consumo. Cuentan con los siguientes campos:

ContractId Identificador del contrato al que pertenecen los datos de uso.

Timestamp Marca temporal del instante en el que se ha producido el evento.

RecordType Tipo de SDR, que en este caso es Quantity.

Concept Nombre del elemento usado.

Value Valor consumido del elemento usado.

Unit Unidad de medida del valor consumido.

Info Información adicional acerca del uso.

En este tarificador, los SDR de cantidades se refieren a usos puntuales (discretos) de un componente en un instante dado. También podrán referirse al incumplimiento de un acuerdo de nivel de servicio en una cantidad determinada. A continuación se muestran dos ejemplos de ficheros SDR de cantidades, tanto de uso puntual como de incumplimiento.

```
{
    "ContractId":"550e8400-e29b-41d4-a716-446655440000",
    "RecordType":"Quantity",
    "Timestamp":"2013-10-29T13:00:00Z",
    "Concept":"Basededatos",
    "Value":2,
    "Unit":"consulta",
     "Info":"AdditionalInfo"
}
```

```
{
    "ContractId":"550e8400-e29b-41d4-a716-446655440000",
    "RecordType":"Quantity",
    "Timestamp":"2013-09-21T21:00:00Z",
    "Concept":"Incumplimiento",
    "Value":2,
    "Unit":"MB/s",
     "Info":"Internet"
}
```

Ficheros SDR temporales

Los ficheros SDR temporales representan usos realizados a lo largo de un periodo de tiempo, indicando el inicio y la finalización del mismo. Cuentan con los siguientes campos:

ContractId Identificador del contrato al que pertenecen los datos de uso.

RecordType Tipo de SDR, que en este caso es Quantity.

StartTime Marca temporal del instante en el que se ha iniciado el evento.

StopTime Marca temporal del instante en el que ha finalizado el evento

Concept Nombre del elemento usado.

Duration Valor consumido del elemento usado.

Unit Unidad de medida del valor consumido.

Info Información adicional acerca del uso.

En este tarificador, los SDR temporales se refieren a usos continuos de un componente entre dos instantes dados. A continuación se muestra un ejemplo de fichero SDR temporal.

```
{
    "ContractId":"550e8400-e29b-41d4-a716-446655440000",
    "RecordType":"Time",
    "StartTime":"2013-10-15T13:00:00Z",
    "StopTime":"2013-10-15T13:01:00Z",
    "Duration":"60",
    "Unit":"segundo",
    "Info":"Internet",
}
```

F

Definición del formato CDR

XSD de definición del formato CDR

A continuación se incluye en código XSD la definición del formato CDR.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="bill">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="fourcaast_id_customer" type="xs:string"/>
        <xs:element name="product_id" type="xs:string"/>
        <xs:element ref="cdr" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="cdr">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="id_service_provider" type="xs:string"/>
        <xs:element name="id_service" type="xs:string"/>
        <xs:element name="time_stamp" type="xs:dateTime"/>
        <xs:element name="id_country" type="xs:string"/>
        <xs:element name="id_event" type="xs:string" minOccurs="0"/>
```

```
        <xs:element name="description" type="xs:string" minOccurs="0"/>
        <xs:element name="cost">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="currency" type="xs:string"/>
              <xs:element name="units" type="xs:double" minOccurs="0"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="id_user" type="xs:string" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Ejemplo de fichero CDR

Puede verse ahora un ejemplo de un pequeño fichero en este formato:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <id>tesla.customers.customer.services.Telefono</id>
    <product_id>20</product_id>
    <cdr>
        <id_service_provider>26</id_service_provider>
        <id_service>Llamada local</id_service>
        <time_stamp>2013-10-28T22:00:00.000</time_stamp>
        <id_country>Spain</id_country>
        <id_event>Pay per use time</id_event>
        <description>2013/10/28 22:00:00 Consumo: 20.0 minutos
        </description>
        <cost>
            <currency>EUR</currency>
            <units>1.0</units>
        </cost>
        <id_user>33</id_user>
    </cdr>
    <cdr>
        <id_service_provider>26</id_service_provider>
        <id_service>Llamada local</id_service>
        <time_stamp>2013-10-28T22:00:00.000</time_stamp>
        <id_country>Spain</id_country>
        <id_event>Pay per use quantity</id_event>
        <description>22:00:00 Consumo: 1 establecimiento</description>
        <cost>
            <currency>EUR</currency>
            <units>0.1</units>
```

```
    </cost>
    <id_user>33</id_user>
</cdr>
```

G

Ejemplo de cancelación

A continuación se muestra un ejemplo de condiciones de cancelación dentro de un contrato.

```
<cancelacion>
    <fijo>20</fijo>
    <precio_periodo>
        <cond>
          <rango>
            <min>1</min>
            <max>4</max>
          </rango>
          <value>20</value>
        </cond>
        <cond>
          <rango>
            <min>5</min>
            <max>10</max>
          </rango>
          <value>10</value>
        </cond>
    </precio_periodo>
</cancelacion>
```

Suponemos que este producto tiene una facturación mensual. Si se realiza una cancelación de un contrato sobre este producto en el tercer mes tras la contratación del mismo, se realizará una facturación temporal de 20€por los meses 3 y 4 y una facturación de 10€por cada mes

restante. Esto hace un total de $2 \cdot 20€ + 6 \cdot 10€ = 100€$. Además, se realiza un cobro fijo de 20€, que es independiente del momento en el que se produzca la cancelación.